ANALYTIC PERFORMANCE MODELING OF CONCURRENT COMPUTER SYSTEMS BY STOCHASTIC PETRI NETS

Scott W. Smart

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

ANALYTIC PERFORMANCE MODELING OF
CONCURRENT COMPUTER SYSTEMS
BY STOCHASTIC PETRI NETS

by

Scott William Smart

June 1981

Thesis Advisor: L. A. Cox, Jr.

Approved for public release; distribution unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Analytic Performance Medeling of Concurrent Computer Systems by Stochastic Petri Nets	5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis: June 1981
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Scott William Smart	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940	12. REPORT DATE: June 1981
	13. NUMBER OF PAGES: 99
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Petri nets, performance evaluation, concurrent computer systems, stochastic analysis, analytic computer models

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Petri nets are presented as a technique for representing computer systems having asychronous, concurrent operation. The structure of the nets are analyzed as a means of demonstrating the correctness of the modeled system. The execution of the petri net is considered as a stochastic process, allowing analysis of the model as a queueing network system by transforming the petri net into its stochastic equivalentnet. It is shown that product form solutions for the state probabilities exist for the class of state machine decompasable nets but for the more general class of consistent petri nets. Solutions for the

DD FORM 1 JAN 73 1473 (Page 1) EDITION OF 1 NOV 68 IS OBSOLETE S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

corresponding open systems are derived by extending the petri net model to include arbitrary sources and sinks.

Approved for public release; distribution unlimited

Analytic Performance Modeling of Concurrent Computer Systems By Stocnastic Petri Nets

by

Scott W. Smart
Lieutenant, United States Navy
B.S., University of Wisconsin--Madison, 1975

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1981

ABSTRACT

Petri nets are presented as a technique for representing computer systems having asynchronous, concurrent operations. The structure of the nets are analyzed as a means of demonstrating the correctness of the modeled system. The execution of the petri net is considered as a stochastic process, allowing analysis of the model as a queueing network system by transforming the petri net into its stochastic equivalent net. It is shown that product form solutions for the state probabilities exist for the class of state machine decomposable nets but not for the more general class of consistent petri nets. Solutions for the corresponding open systems are derived by extending the petri net model to include arbitrary sources and sinks.

TABLE OF CONTENTS

I. INTRODUCTION ---------- 6

A. PETRI NETS AS A PERFORMANCE MODEL ----------11

B. REVIEW OF RELATED WORK ----------21

C. OUTLINE OF SUCCEEDING SECTIONS ----------25

II. PETRI NET THEORY ----------27

A. VECTOR ADDITION SYSTEMS ----------36

B. SUB-CLASSES OF MARKED PETRI NETS ----------44

III. STOCHASTIC PETRI NETS ----------62

A. TIMED EVENTS IN PETRI NETS ----------62

B. MARKOV ANALYSIS OF PETRI NETS ----------65

C. CLOSED PETRI NET SYSTEMS ----------71

D. OPEN PETRI NET SYSTEMS ----------82

E. CONSISTENT PETRI NET SYSTEMS ----------87

IV. CONCLUSIONS ----------90

APPENDIX LIST OF NOTATION ----------93

LIST OF REFERENCES ----------95

INITIAL DISTRIBUTION LIST ----------99

I. INTRODUCTION

With the explosion in numbers and types of computer hardware components witnessed in the past few years, computer system design has become an extremely complex task. Cox has observed [1]:

> "Today's computers are among the most complex man made systems in existence today. The development of such systems represents a significant commitment of physical and mental resources. This cost can only be justified if these computing devices serve their intended purpose -- the efficient processing of data in response to specific needs."

If we are to make effective use of these development resources, it is necessary to provide the system designer with tools which allow him or her to create and analyze sophisticated system designs.

Several trends have emerged which have accelerated the search for new and better design tools. The low cost of LSI and VLSI hardware components coupled with technological advances in digital communications has led to the evolution of a wide variety of multiprocessor systems, distributed computing applications, and computer networks. As early as 1976, Anderson and Jenson [2] identified 27 different networking schemes being used in prototype or actual systems. When one considers the number of communication protocols and transmission media which have been proposed or

implemented, the number of variables involved in the hardware design process becomes truly awesome.

Likewise, innovations in system software have given rise to such concepts as distributed operating systems, on-line data bases, and interactive programming, to name a few. Kobayashi and Konheim [3] have noted that:

> "With the increasing complexity and sophistication of computer communication systems, modeling and performance evaluation [emphasis their's] are becoming critical issues in the design and operation of such systems. It is apparent that for a cost-effective design we must be equipped with systematic methods of predicting quantitative relations between system resource parameters, system workloads, and measures of system performance."

Several characteristics of these systems may be identified which determine what types of models are appropriate for performance evaluation. A central concept is that of concurrent or parallel processing. Each node in a network, or processor in a multiprocessing system, is capable of independent computation. At the same time, system or global resources such as memory and communication links must be shared by the various processing elements. This results in enforced cooperation between otherwise independent processes.

Since most resources in computing systems tend to be scarce in relation to the demands on them, contention exists between resource users which must be arbitrated. This problem is complicated by the observation that demands in computer systems are not constant. For example, in computer

communication networks empirical evidence shows that demands tend to be bursty and sporadic. As a result, resource demands must be viewed as stochastic in nature and system models must be capable of expressing the probabilistic elements of the modeled system.

The uncertainty in resource demand results in two phenomena which must be considered in performance modeling. The first is the creation of queues of users which require service but must wait for resource availability. The second, and interrelated, phenomenon is the delay which users experience while waiting for resources and while being served. The field of queueing theory has attempted to answer these questions and others concerning the probabilistic properties of systems in both analytic and simulation models.

One purpose of performance prediction modeling is to analyze system designs in terms of performance measures or indices. Ferrari [4] has identified three performance index classes: productivity, responsiveness, and utilization. A number of specific measurements may be computed from the model to express these indices -- throughput, waiting time, and utilization for example.

A second purpose of performance modeling is to verify proper system operation. It is important to ensure that the underlying system is deadlock-free, or at least to predict the circumstances under which deadlock could occur. In

addition, it may be necessary to demonstrate that the system enforces synchronization or mutual exclusion among elements of the system.

It is not sufficient for a model to be capable of providing answers to these questions. The model must also be amenable to validation; that is, the determination of how accurately the predicted results conform to the modeled system. This may be particularly difficult to accomplish when the actual system does not exist or is otherwise not available. Finally, it is desirable that the model be robust in order that its domain of validity extend over as large a range of systems as possible.

Performance models can be divided into two separate types -- simulation and analytical. A simulation may take various forms; however, the form most often associated with performance modeling is computer based, discrete event simulation. This model consists of a program which describes the state of the modeled system in terms of system entities and their attributes at each point in time. Attributes are varied as a result of the instantaneous occurrence of events in the system. The model then tracks the changes in attributes over time to determine the required performance measures. Simulations suffer from several shortcomings. To accurately model the system, the programs must be complex resulting in a significant software engineering problem; the model must be carefully designed and verified to ensure

proper operation. Because of the probabilistic nature of the system, it is necessary to operate the simulation for numerous runs and over a large span of simuation time to ensure statistically meaningful results. This can lead to significant computing costs for the simulation. Finally, if the system parameters are to be changed, or elements of the system altered, it is necessary to change the simulation and rerun it. Once again, the costs of software modification and model operation must be met. This can make it difficult to generalize or abstract from the simulation.

An alternative to simulation modeling is analytic modeling. In this method the system is expressed as a set of mathematical equations. Determining the performance measures for the system amounts to finding the appropriate solutions to the system equations. Unfortunately, in many cases the solutions are mathematically intractable or computationally inefficient and require that simplifying assumptions be made about the system. However, to quote from Kobayashi and Konheim once again:

> "Even when a decision is made for simulation, an analytic solution, however crude it may be, can serve as a guideline in narrowing down a range of system configurations and parameters under which a simulation runs. It also could save a considerable amount of modeling efforts, by detecting possible errors introduced in the design and implementation phases of a simulation."

This thesis investigates an approach to the analytic modeling of computer systems based on using a

graph-theoretic technique -- petri nets -- as a representation for system elements and their interactions. By analyzing the structure of a petri net model, it is possible to answer a number of questions regarding the operation of the modeled system. We show that it is possible to model the stochastic nature of computer systems by extending the petri net model to allow nondeterminism in the net to be expressed in terms of probability distributions. It is then possible to consider a petri net as an analog to a queueing network system and therefore it is possible to apply the known methods of Markov analysis to the nets to obtain analytic solutions for the system. The problem we address in this thesis is the evaluation of nondeterministic and stochastic petri nets using queueing theory techniques. The solutions which result combine the structural properties of petri nets with the capability for performance prediction.

A. PETRI NETS AS A PERFORMANCE MODEL

Consider a simple computer system consisting of two cooperating, concurrent processes A and B running on separate hardware. Process A is a producer, and process B is a consumer of data. It is desired that a handshake protocol be implemented between the two processes (see Figure 1.1). How might this be represented?

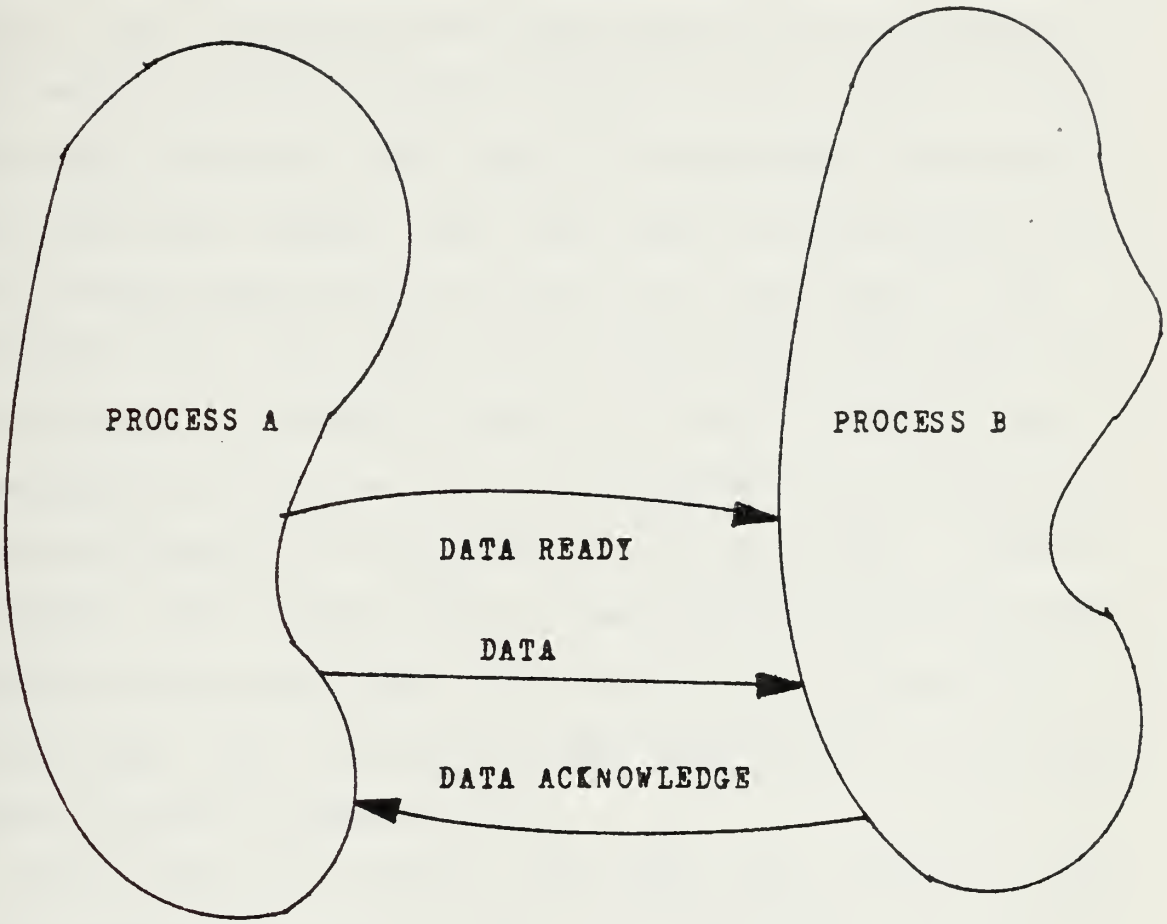


Figure 1.1 Communicating Processes

One possibility is to list the events which must occur during the handshake process. Using this event list, we could then determine what conditions must be satisfied for each event to occur, and what changes in the system result from that occurrence. This could be graphically displayed in the following manner. Let the events be represented by a bar, and the conditions by circles. For each event, draw a directed arc from each circle (condition) which must hold for the event to occur to the bar (event). Next, draw a directed arc from the bar to each circle (condition) which holds as a result of the occurrence of the event. Finally, determine the initial state of the system by deciding which conditions initially hold, and place a dot (token) in a circle for each holding of that condition. The resulting graph is shown in Figure 1.2.

This graph is called the petri net model of the communication protocol, after C. A. Petri who first studied them in the 1960's [5]. Note that each event may have one or more input conditions, and one or more output conditions. In the language of petri nets, the events are called transitions and the conditions are called places. The net operates by moving tokens around the net in accordance with the firing rule, which states that an event may occur when each input place to the event has a token assigned to it. The transition fires by removing a token from each input place and depositing one in each output place. The state of

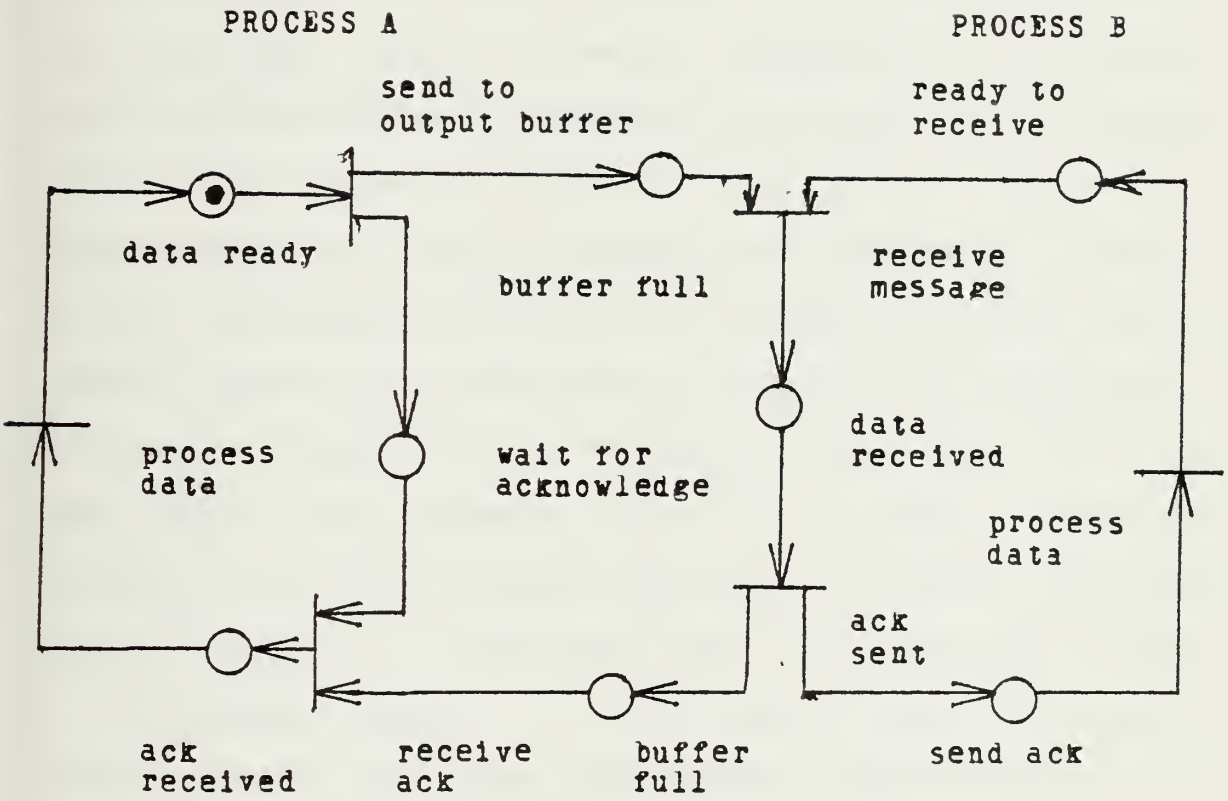


Figure 1.2 Petri net model of a communication protocol

each place is determined by the number of tokens in it. The system state, or marking, is then represented by a vector whose elements are the state of each place in the net.

One feature of the petri net model is that the tokens can represent either control flow or data flow; the difference purely lies in how the net is interpreted by the designer. This has led to some difficulty in modeling data-dependent events. Several attempts have been made to overcome this problem by extending the definition to include specialized places such as conditional places, which cause transitions to fire in different ways depending on whether or not the condition place holds a token. In either case the token content of the conditional place is not changed by the firing. An example of this extension is the Macro E-Net Noe and Nutt have proposed [6]. In a slightly different extension, inhibitor places were added to the net [7] which prevent firing of a transition when the place holds a token.

A second feature of petri nets is that the firing of transitions is inherently asynchronous and concurrent -- for example in Figure 1.2 the <process data> events for both processes can fire independently of each other. When necessary, the operation of concurrent processes can be coordinated through the use of multiple input places; the <receive message> event is an example of this type of interaction.

The basic petri net, as depicted here, does not attempt to model the time required for execution. Extensions to the theory which account for execution times were investigated by Ramchandani [8] and others. This technique allows for analysis of the time-related operation of the net.

1. Analysis of petri nets

Petri nets may be analyzed in a number of ways. Much of the work, particularly at MIT, has been based on using techniques from automata theory and investigating the nets as formal language generators [9, 10]. Using this method, Hack has proven a number of decidability questions about the possible configurations (system states) of the net. Other work [11,12,13] has emphasized the graph theoretical properties of the nets in analyzing their structure. This methodology has led to the identification of several subclasses of petri nets based on special structural characteristics.

Simulations based on petri net models have been widely used in analyzing hardware designs of concurrent systems and data flow computing. Typically, the system to be modeled is reduced to its petri net equivalent which then serves as input to the simulator. The simulator then executes the petri net in much the same manner as a conventional discrete event simulation.

2. Uses and Limitations of the Petri Net Approach

Petri nets have been used to model a large number of concurrent software and hardware systems. In hardware design, they have been used as a basis for developing speed independent logic [14,15] by proving the conditions for which circuits are free of races when operating in fundamental mode. More ambitious applications have involved the analysis of multiprocessor systems such as the CDC 6600 [16,17], IBM 360/91 [18], Amdahl 470 V/6 [17], and US Navy SEAFIRE weapons system [19]. Most of these applications have involved simulation modeling due to the complexity of the designs. A slightly different approach to hardware design has been the design by step-wise refinement method. Valette [20] has shown that single transitions could be replaced by more complex structures when certain conditions were met. Using this method, each component of the system can be separately analyzed and formed into an independent structure he called a well-formed block. By substituting these blocks for transitions in the net, it is possible to retain the properties of the original net. This hierarchical structure also simplifies the problem of understanding the operation of the net. Figure 1.3 shows how this might be accomplished.

Petri net models have been used in the analysis of software designs as well. For example, they have been used to verify the correctness of communication protocols [21]. Operating system synchronization primatives such as P and V

[22] may be modeled using simple net structures. Figure 1.4 gives an example of how mutual exclusion could be represented. The contents of place S is the semaphore value for the protected resource (in this case 1). Each process which requires exclusive use of the resource has an event P(S) which may fire only if the semaphore is non-zero. Firing of the corresponding V(S) event returns the token to the semaphore. The significant advantage of using a petri net is that mutual exclusion can be proven by showing that only one P(S) event can be fired until the corresponding V(S) event occurs.

Lest the reader get the impression that petri nets are the ultimate modeling tool, it should be noted that petri nets have limitations in their modeling ability. The lack of a mechanism for handling data dependent events as described earlier has made it difficult to model actual systems using deterministic nets. A second related limitation is the use of deterministic transition firing times in the analysis of timed nets. In our opinion, the restriction of petri nets to deterministic modeling has been responsible for the lack of attention given to them in recent years.

Our view of petri nets emphasises the non-deterministic modeling capabilities of the nets. We consider the transitions in the net to be service centers which operate according to some service time distribution. In this approach, the places can represent queues of tokens awaiting

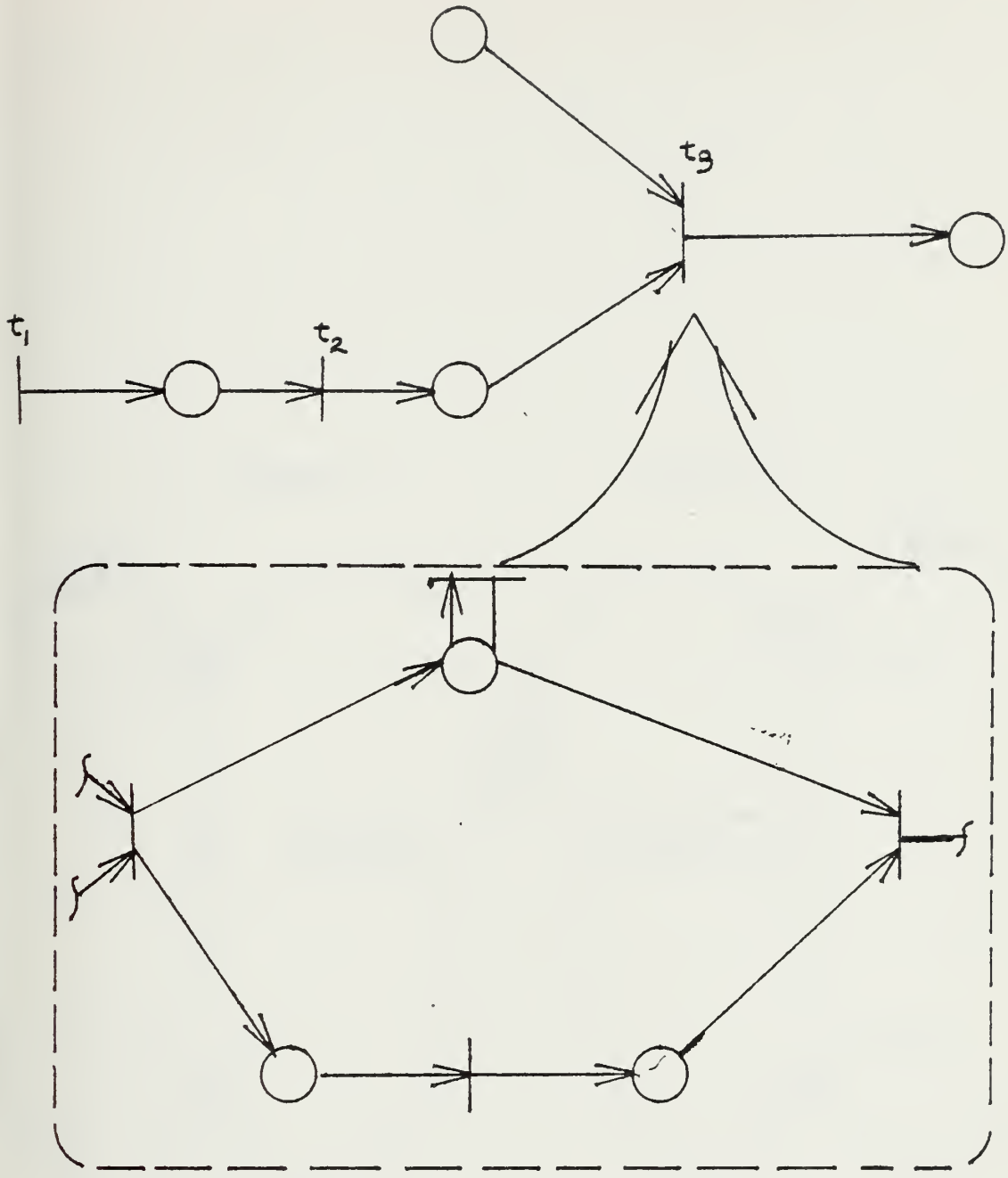


Figure 1.3 Hierarchical modeling in Petri Nets. Here, transition t3 is replaced by the subnet N (from Peterson [23]).

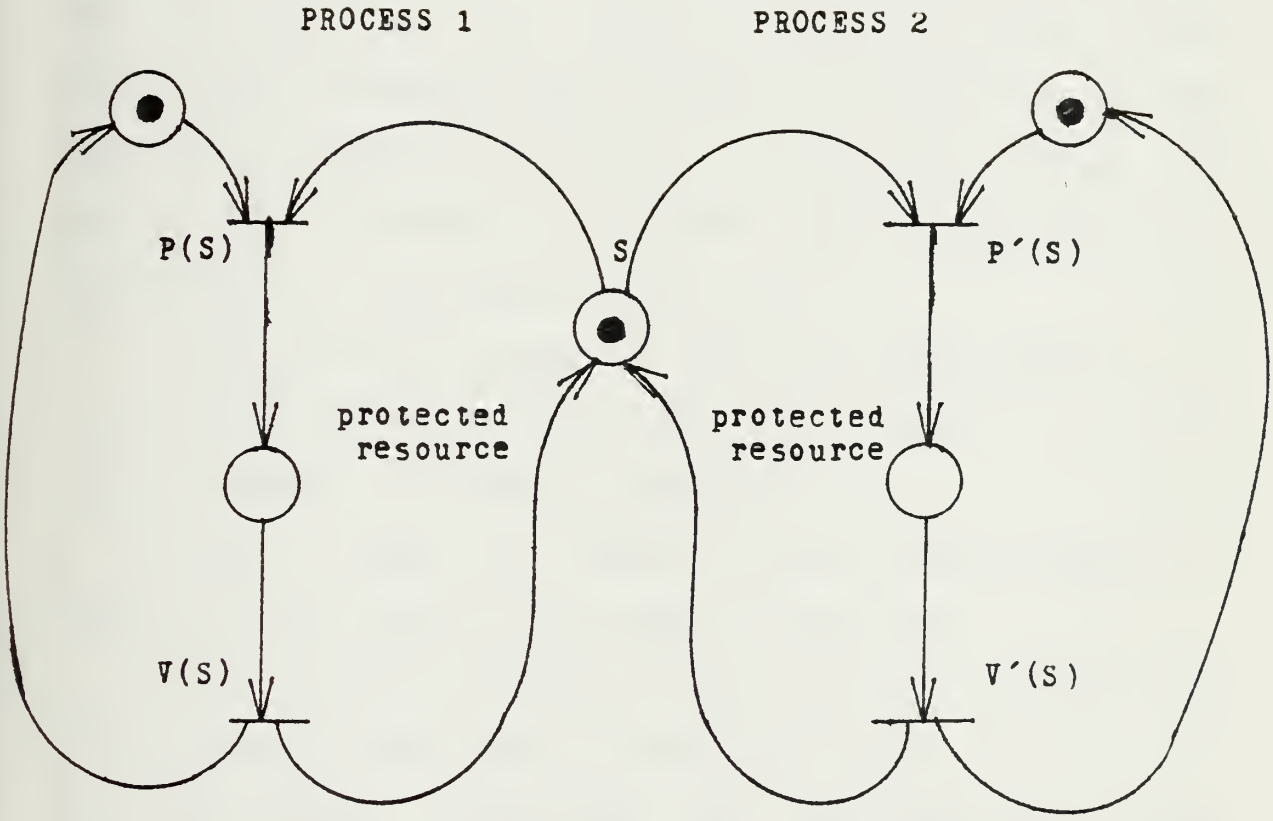


Figure 1.4 Petri net modeling of mutual exclusion using Dijkstra's P and V (from Peterson [23]).

service. By regarding nets in this manner, the known results of queueing network theory may be applied to the analysis while maintaining the benefits of the underlying structure of the nets. We feel that this will result in a better modeling tool for analyzing concurrent systems.

B. REVIEW OF RELATED WORK

The basis for concurrent system modeling has largely been derived from two papers by Karp and Miller [24,25]. These papers proposed several models for concurrent systems. Computation graphs are similar to petri nets except that places were modeled as directed arcs between events and labeled with a four-tuple defined as:

A -- the initial number of words in a FIFO queue

U -- the number of words added to the queue as a result of the firing of the input transition

W -- the number of words removed from the queue as a result of the firing of the output transition

T -- a threshold number of words (perhaps greater than W) required for the output transition to fire.

Karp and Miller established the requirements for several important concepts including liveness and boundedness which we discuss in greater detail later.

Reiter [26] extended computation graphs by adding a fifth element tau to the directed arc labeling which represented the execution time required for the output

transition to process the W data words when firing. Reiter then determined a method for finding possible sequences of firings of transitions. For cyclic graphs, he found a lower bound for the cycle period of the graph. Computation graphs have been used in the performance analysis of data flow processors [27].

The second paper by Karp and Miller [25] investigated parallel program schemata and vector addition systems. A parallel program schema modeled parallelism in programs by establishing computation states and rules for state transitions. The concept of FORK and JOIN which have been applied to data flow and other MIMD architectures was developed to express the creation of concurrent processes from a sequential process, and the combination of concurrent processes into a sequential process. The equivalence of petri nets, computation graphs, and parallel program schemata was shown by Miller [28].

Most of the research on petri nets in the United States has been conducted at MIT. Commoner and Holt [11,12] studied a deterministic subclass of petri nets known as marked graphs and proved several theorems regarding their system state spaces. A more general subclass of deterministic nets, persistent nets, was studied by Landweber and Robertson [13] who proved that the theorems regarding marked graphs were applicable to this class. A final deterministic subclass referred to as state machine decomposable was investigated

by Ramchandani. In addition, Ramchandani analyzed this subclass with deterministic transition firing times to derive expressions for the minimum net cycle period in a manner analogous to that Reiter used for computation graphs.

Other classes of petri nets have been defined which improve the tractability of the problem of finding solutions to performance measures while retaining characteristics which are desired in the modeled systems. In particular, live, bounded, conservative, and consistent nets have received the most attention. The properties of these classes will be examined in detail in the next section.

Our work extends this previous work by focusing on the broader class of nondeterministic, consistent petri nets which appears to be the most general class for which solutions to performance questions may be found. While nondeterminism increases the difficulty of dealing with petri nets analytically, much of the previous work on petri net structure remains pertinent to this class. Nondeterminism allows the net to represent different types of data, for example, by regarding the data type or transition firing time as a random variable. To analyze the performance of nondeterministic petri nets, we consider them as analogs to queueing network models. The basis for analysis of these models is the classic work of Jackson [29] for the case of open systems, and Gordon and Newell [30] for closed systems. Both of these models relied upon poisson

arrival processes, first come, first serve (FCFS) queueing discipline, and exponential service departure processes to ensure that the Markov property was met and allowed the system state probabilities to be expressed as the product of the marginal state probabilities. This quality is known as the product form solution and is required for computationally efficient solutions. Most of the recent work in queuing network theory has attempted to find product form solutions for more general systems. Jackson [31] considered systems where the arrival rates and service rates were functions of the queue lengths (states) at the various nodes. Baskett et al. [32] extended this to open and closed networks where customers were of different classes and the queueing discipline was FCFS, no queueing, and last come, first serve with preemptive resume (LCFS-PR). All these cases were shown to have product form solutions. In addition, they showed that a condition known as local balance was a necessary condition for product form solutions.

Chandy et al. [33] considered the question of local balance in more detail and developed the more general notion of station balance which was also shown to be necessary and sufficient for product form solutions if non-exponential differentiable service distributions were used. In addition they proved that arbitrary service distributions satisfied station balance if processor sharing (PS) or LCFS-PR

disciplines were used. In the case of exponential service time distributions for all service classes, they showed that FCFS and priority disciplines resulted in station balance being met. This result has been extended to any work conserving discipline.

In summary, it has been shown that product form solutions exist for queueing networks with the following properties:

1. For exponential service, any work conserving queueing discipline may be used.

2. For PS and LCFS-PR disciplines, any differentiable service distribution may be used.

3. The solutions do not depend on the routing used for the customers.

C. OUTLINE OF SUCCEEDING SECTIONS

In Section II, we present the formal definitions of petri nets and derive several useful properties. The state space -- the set of system states the net may occupy -- is determined by constructing the reachability set of the petri net. The conditions under which the state space is finite and recurrent are then considered. It is shown that nets with certain structural characteristics give rise to these restricted state spaces. In particular, the classes of marked graphs, state machines, and consistent nets are considered because of their relation to realizable systems.

In Section III we turn to nets with timed events. The case of deterministic routing and transition time is first considered. Next, the (probabilistic) class of state machines with nondeterministic routing and exponential transition times is examined. It is shown that for the class of state machine decomposable nets it is possible to transform the net into a stochastic equivalent net which is analogous to a closed queueing network. For this class of nets a product form solution for the state probabilities is derived.

Next, petri nets which allow external sinks and sources are defined. Again, it is shown that the stochastic equivalent nets may be analyzed as queueing networks; in this case an open system.

Finally, the class of consistent petri nets with exponential firing times is considered and it is shown that product form does not exist for this class of nets.

II. PETRI NET THEORY

In this section the relevant petri net theory is presented. Much of the work follows that of Commoner and Holt [11], Kraft and Miller [25], and Ramchandani [8]. The notation used is primarily that of Peterson [23].

The petri net was defined informally in the preceding section as a means for representing related events and their conditions in systems. We now formalize this notion by defining the petri net N and its directed graph representation.

Definition 2.1 $N = \langle P, T, I, O \rangle$

where

$P = \{p_i \mid p_i$ is a place in the net$\}$

$T = \{t_j \mid t_j$ is a transition in the net$\}$

$I : (P \times T) \rightarrow \mathcal{N}$ such that if p_i is an input place to t_j, $I(p_i, t_j) \geq 1$ and $\emptyset$ otherwise

$O : (P \times T) \rightarrow \mathcal{N}$ such that if p_i is an output place to t_j, $O(p_i, t_j) \geq 1$ and $\emptyset$ otherwise

with the requirement that $\forall t_i \in T \quad \exists p_j, p_k \in P \mid I(p_j, t_i) \neq \emptyset \wedge O(p_k, t_i) \neq \emptyset$ and $\forall p_j \in P$

$I(p_j, t_i) > \emptyset \implies O(p_j, t_i) = \emptyset$

We further define the following sets:

$\bullet t_i = \{p_j \mid I(p_j, t_i) > \emptyset\}$

$t_i^\bullet = \{p_j \mid O(p_j, t_i) > 0\}$ where $t_i \in T \wedge p_j \in P$

and likewise:

$^\bullet p_j = \{t_i \mid O(p_j, t_i) > 0\}$

$p_j^\bullet = \{t_i \mid I(p_j, t_i) > 0\}$ where $t_i \in T \wedge p_j \in P$

These sets are referred to as the input and output sets, respectively. Note that the definitions of the functions I and O require that:

$$\forall t_i \in T, \ {}^\bullet t_i \neq \phi \wedge t_i^\bullet \neq \phi \wedge {}^\bullet t_i \cup t_i^\bullet = \phi$$

The set of places represents the conditions in our informal model, and the set of transitions represents the events. The input and output functions specify the preconditions for an event to occur and the results of the event occurrence.

Corresponding to each petri net we define a bipartite, directed graph as follows:

$\forall p_i$ $p_i \in P$ draw a circle representing a place

$\forall t_i$ $t_i \in T$ draw a vertical bar representing a transition

$\forall t_i, p_j$ if $p_j \in {}^\bullet t_i$ draw a directed arc from p_j to t_i

$\forall t_i, p_j$ if $p_j \in t_i^\bullet$ draw a directed arc from t_i to p_j

We will use the notions of petri net and petri net graph interchangibly (since they are equivalent).

Example 1.

Consider the following petri net $N = \langle P, T, I, O \rangle$ with

$P = \{ p_1, p_2, p_3, p_4, p_5, p_6, p_7 \}$

$T = \{ t_1, t_2, t_3, t_4, t_5 \}$

let $n = |P|$ and $m = |T|$. Then I and O may be

represented by the n X m incidence matrices:

$$C_I = \begin{vmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{vmatrix} \qquad C_O = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{vmatrix}$$

where $C_{I\ i,j} = I(p_i, t_j)$

and $C_{O\ i,j} = O(p_i, t_j)$

The corresponding petri net graph is shown in Figure 2.1.

If we associate with each place in a petri net N a non-negative integer marking function μ, we have the following definition of a marked petri net M:

Definition 2.2 A <u>marked petri net</u> $M = \langle P, T, I, O, \mu \rangle$
where P,T,I,O are defined as before and
μ: P --> N

Each function μ_K defines a marking of the net. In graph notation the petri net graph is extended to a marked petri net graph by adding tokens to places as follows:

$\forall p_i \in P$, if $\mu_K(p_i) = n$ then place n tokens (dots) on place p_i.

Figure 2.2 shows a possible marking for the petri net of Example 1. Clearly a (countably) infinite number of possible markings exist for any petri net. This marking represents the (possibly multiple) holdings of conditions at

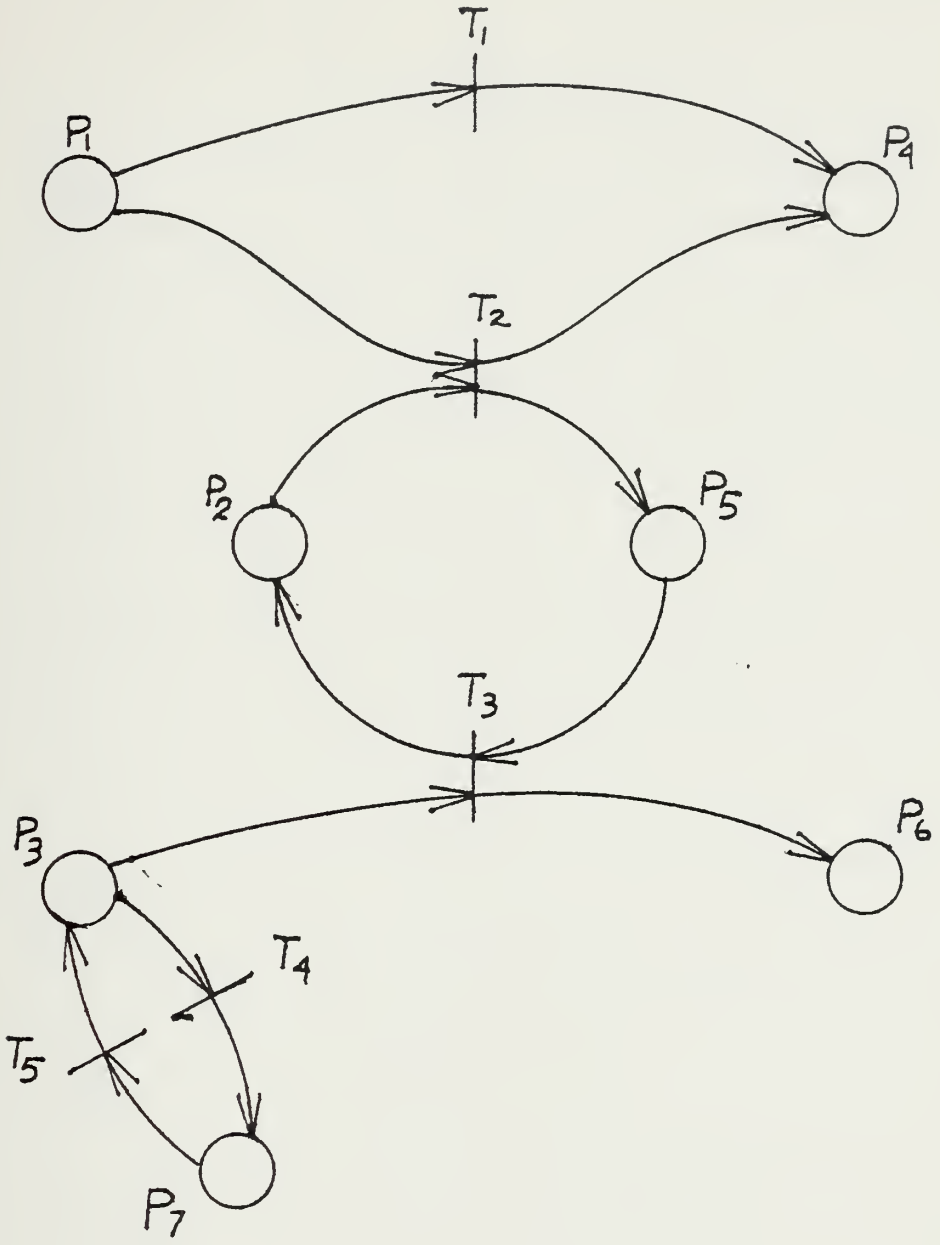


Figure 2.1. Petri net graph

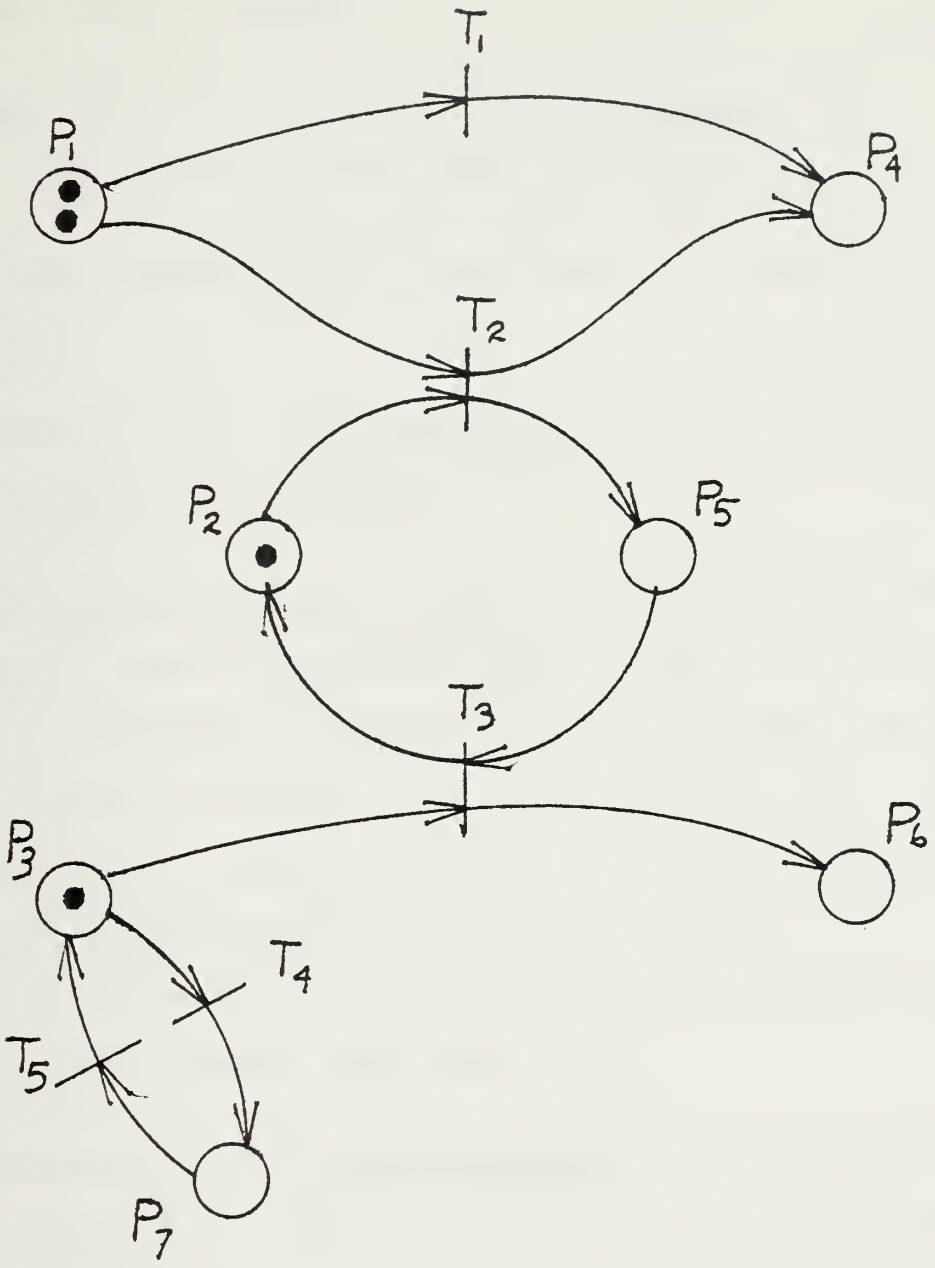


Figure 2.2. Marked Petri net graph

some instant of execution and may be viewed as a description of the state of the net.

Definition 2.3 A transition t is <u>enabled</u> iff

$$\forall p_j \quad p_j \in {}^\bullet t_i \quad \Longrightarrow \quad \mu(p_j) \geq I(p_j, t_i)$$

For each marking μ_k we can define the enabling set S_k as follows:

Definition 2.4 The <u>enabling set</u> $S_k \subseteq T = \{t_i \mid t_i$ is enabled by marking $\mu_k\}$.

In our graph notation, a transition is enabled if each arc directed into the transition has a token in its originating place. Referring to Figure 2.2, it will be seen that t_1, t_2, and t_4 are enabled and therefore $S_k = \{t_1, t_2, t_4\}$. Transition t_3 is not enabled since there is no token in place p_5 corresponding to the arc $p_5 \rightarrow t_3$.

An enabled transition may fire to create a new marking. That is, we define a function:

Definition 2.5 A <u>firing function</u> for a marked net M is

$$F : \mu \times S \rightarrow \mu \quad \text{such that } F(\mu_k(p_j), t_i) = \mu_k(p_j) - I(p_j, t_i) + O(p_j, t_i) = \mu_{k+1}$$

where $p_j \in P \ \wedge \ t_i \in S_k$.

F is defined for all markings for which the set S_k is non-empty. Continuing with the vector notation introduced earlier for the functions I and O, we may denote each

marking μ_k as an n-element vector U_k where $n = |P|$ and $U_k(i) = \mu_k(p_i)$ for $i = 1,2,\ldots,n$. Then F may be expressed as:

$F(U_k, t_i) = U_k - C_{Ii} + C_{Oi}$ where C_{Ii} and C_{Oi} are the ith columns of the input and output incidence matrices respectively.

Definition 2.6 Marking μ_l is <u>directly reacnable</u> from μ_k if $(t_i \in S_k) \wedge F(\mu_k, t_i) = \mu_l$ and is denoted as: $\mu_k \xrightarrow{t_i} \mu_l$.

Returning to our earlier example, transition t_2 may be fired (recall that the enabling set $S_k = \{t_1, t_2, t_4\}$). By inspection of Figure 2.2, we specify the vector for marking μ_k:

$$U_k = \begin{vmatrix} 2 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{vmatrix}$$

The resulting marking μ_{k+1} =

$$U_{k+1} = U_k - C_{I2} + C_{O2} = \begin{vmatrix} 2 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{vmatrix} - \begin{vmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{vmatrix} + \begin{vmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{vmatrix} = \begin{vmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{vmatrix}$$

In graph notation, one token is removed from each place having an arc directed into the firing transition, and one token is added to each place having an arc directed from the

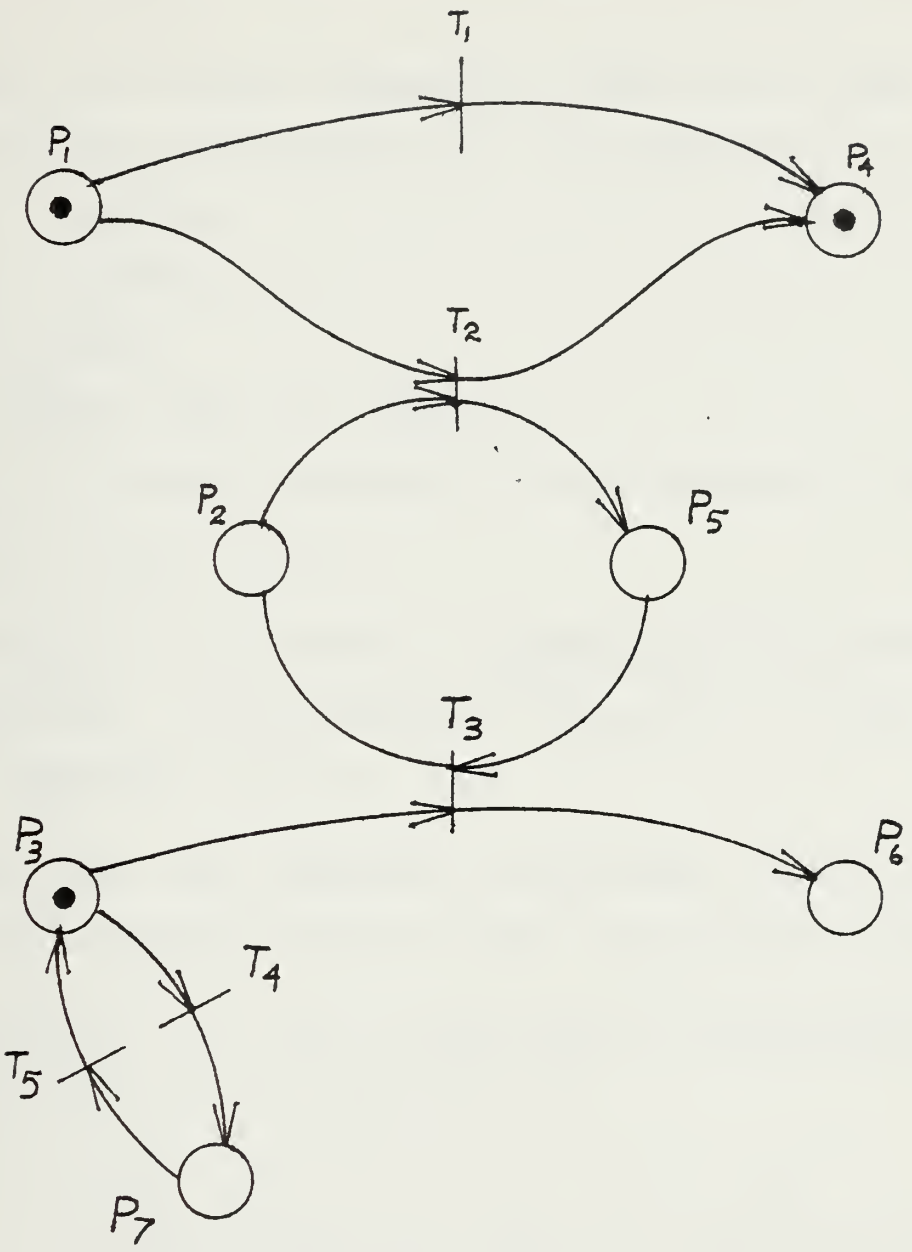


Figure 2.3. Petri net graph after firing transition t2

firing transition. The resulting marking is shown in Figure 2.3.

Next we consider sequences of transition firings. Assume the following firings exist for some marked net M:

$$\mu_0 \overset{t_i}{\Rightarrow} \mu_1$$

$$\mu_1 \overset{t_j}{\Rightarrow} \mu_2$$

Then we denote

$$\mu_0 \overset{t_i}{\Rightarrow} \mu_1 \overset{t_j}{\Rightarrow} \mu_2 \quad \text{or} \quad \mu_0 \overset{\sigma}{\Rightarrow} \mu_2$$

where the firing sequence σ is some sequence of transition firings $(t_1, t_2, \ldots, t_n)$.

Definition 2.7 Marking is <u>reachable</u> from marking μ_k iff

$$\mu_k \overset{\sigma}{\Rightarrow} \mu_1 \iff \exists \sigma \mid \mu_k \overset{t_1}{\Rightarrow} \mu_{k+1} \overset{t_2}{\Rightarrow} \mu_{k+2} \overset{t_3}{\Rightarrow} \cdots \overset{t_n}{\Rightarrow} \mu_1$$

where $\sigma = (t_1, t_2, \ldots, t_n)$

If we form the reflexive, transitive closure of the transition firing relation we have the following definition:

Definition 2.8 Given a marked net M, the reachability set Q(M) is defined inductively as:

basis: $\mu_0 \in Q$

induction: $\mu_k \in Q \wedge \exists t_i \in T \mid \mu_k \overset{t_i}{\Rightarrow} \mu_{k+1} \Longrightarrow \mu_{k+1} \in Q$

We are interested in determining the elements of the set Q(M). To do so, we must first formalize the previously introduced vector notation as a vector addition system.

A. VECTOR ADDITION SYSTEMS

The concept of vector addition systems was first explored by Karp and Miller [25]. This section is largely derived from their work.

A vector addition system is defined as a pair $V = (d,W)$ where d is an r-dimensional vector with $i \in N$ and W is a finite set of r-dimensional vectors $w_1, w_2, \ldots, w_n$ with $w_j(i) \in I$. The reacnability set R(V) for a vector addition system is the set of vectors composed by adding elements of the set W to d. That is

$$R(V) = \{x \mid x = d + (w_1 + w_2 + \ldots + w_n)^* \wedge x(i) \geq 0\}.$$

In addition, the following terminology is used:

The relation $\leq$ is defined as

$x \leq y \iff$ for $i = 1,2,\ldots,r$ $x(i) \leq y(i)$

The symbol ω is defined such that if n is any integer, $\omega > n$ and $n + \omega = \omega$.

The reachability set R(V) can be determined by constructing the corresponding reachability tree T(V). Nodes in the tree consist of r-dimensional vectors x,y,z,... with $x(i) \in N \cup \{\omega\}$, for $i = 1,2,\ldots,r$. The relation $\prec$ is defined as $x \prec y \iff$ a directed path exists from x to y in T(V). Let d be the root of the tree. Descendants are constructed recursively as follows:

1. If $x \prec y$ and $x = y$, y is a leaf node.

2. Otherwise, construct successor nodes to y with vectors $y + w_1$, $y + w_2$, ... , $y + w_n$ where for i =

$1,2,\ldots,r$ $y(i) + w_j(i) \geq 0$. In addition, if there exists a node z such that $z \prec y$ and $z \leq y + w_i$ and $z(j) < y(j) + w_i(j)$ for some j, then $y(j) + w_i(j) = \omega$.

To illustrate the construction of this tree consider the following example from [25]:

Example 2.

Let V = (d,W) where

d = [1,0,0,0,0]

W = { [-1,1,0,0,0] , [-1,0,0,1,0] , [0,-1,2,0,0] ,
[0,1,-1,0,0] , [0,0,0,-1,2] , [0,0,0,1,-1] }

The resulting reachability tree is shown in Figure 2.4.

The following theorems from [25] allow us to answer decidability questions about vector addition systems by inspecting the reachability tree.

Theorem 2.1

$$\forall x \exists y \mid y \in R(V) \wedge (x \leq y) \Leftrightarrow z \mid z \in T(V) \wedge (x \leq z)$$

The proof, while straight foward, is rather lengthy and so is not included here, but may be found in [25].

Theorem 2.2 For any vector addition system V, T(V) is finite.

We first show the following two lemmas:

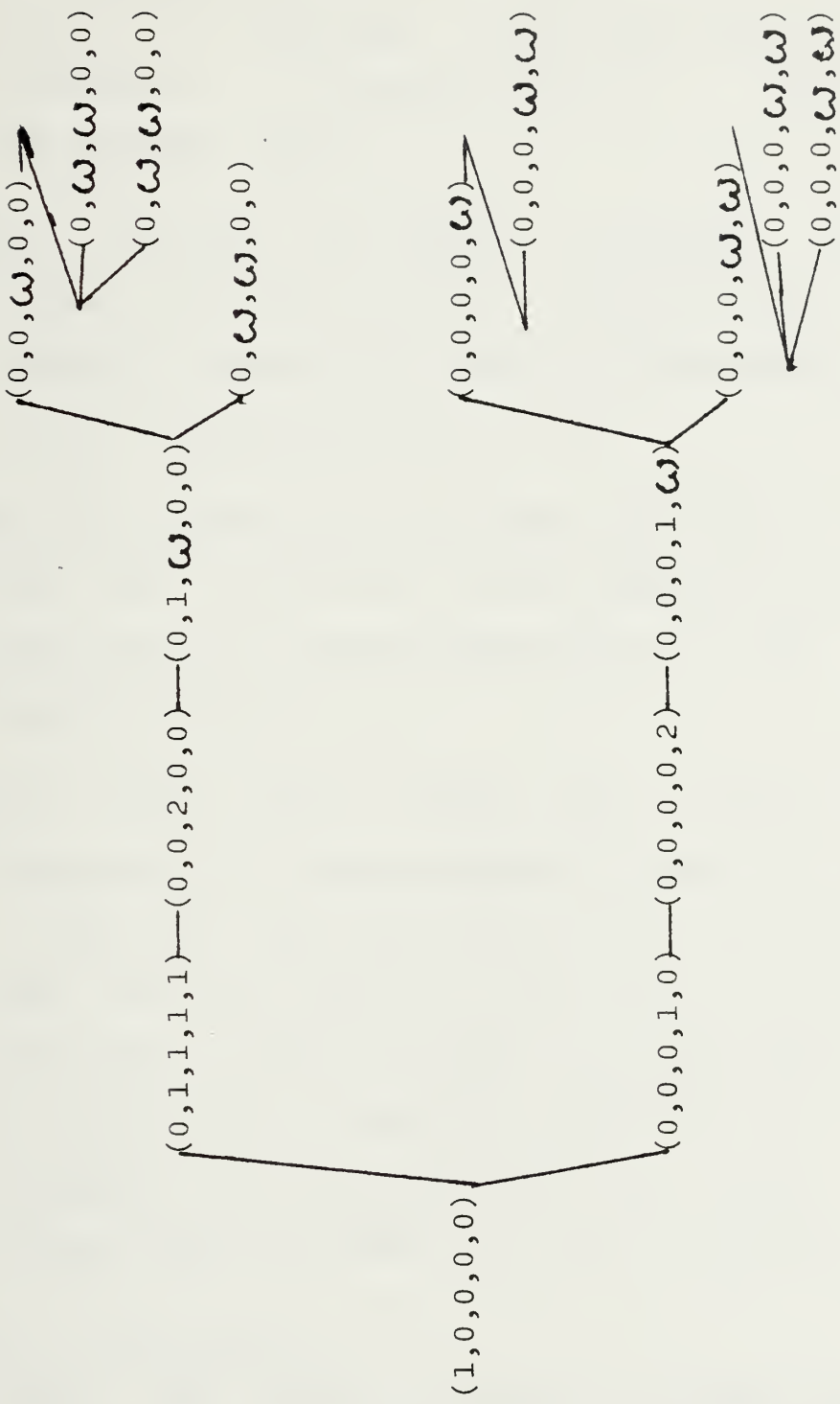


Figure 2.4. Reachability tree for Example 2.

Lemma 2.2.1

Let $p_1, p_2, p_3, \ldots, p_n, \ldots$ be an infinite sequence of elements of $(N \times \omega)$. Then there exists an infinite subsequence $p_a, p_b, p_c, \ldots, p_z, \ldots$ such that $p_a \leq p_b \leq p_c \leq \ldots p_z \leq \ldots$.

Proof. Construct an infinite subsequence by selecting elements with first entries nondecreasing. From this sequence, construct an infinite subsequence with second entries nondecreasing, and so on.

Lemma 2.2.2 (König Infinity Lemma) Let T be a tree such that each vertex has a finite number of successors and there exists no infinite directed path from the root. Then T is finite.

Proof. Since each vertex has a finite number of successors, let n be the maximum number of successors for any node. Then there are at most n paths of length 1 from the root, and if p is some node in the tree, there are at most n paths of length 1 in the subtree having p as its root. Since no infinite path exists by assumption, let m be the maximum path length. Then the maximum number of nodes is n^m and T is finite.

Proof of theorem 2.2.

Assume there exists an infinite directed path from the root of T(V) with node sequence $p_1, p_2, p_3, \ldots, p_n, \ldots$.

Then by lemma 2.2.1, there exists a subsequence $p_a, p_b, p_c, \ldots, p_z, \ldots$ with $p_a \leq p_b \leq p_c \leq \ldots p_z \leq \ldots$. By definition, if $p_a = p_b$ then p_b is a leaf node and therefore the inequality is strict and $p_a < p_b < p_c \ldots < p_z < \ldots$. Again by definition, since $p_a < p_b$, p_b must have one more entry equal to ω than p_a. Since the number of entries is finite, this is impossible and therefore no such infinite directed path exists. By lemma 2.2.2, T(V) is finite.

Using these theorems, it is possible to prove the following decidability theorems about the reachability set R(V):

Theorem 2.3 Given some finite $n \in \mathcal{N}$,
$\forall x \; x \in R(V) \wedge (x(i) \leq n)$ is decidable.

Proof. Construct T(V). By theorem 2.2 T(V) is finite. Therefore $\forall y \; y \in T(V) \wedge (y(i) \leq n)$ is decidable. By theorem 2.1 then, the question is decidable for R(V).

Theorem 2.4 Given some subset of the entries for the r- dimensional vector addition system $\Theta \subseteq \{1,2,\ldots r\}$,
$\forall x \; x \in \mathcal{N}^r \; \exists y \; y \in R(V) \wedge (\forall i \; i \in \Theta)$,
$y(i) \geq x(i)$ is decidable.

Proof. By the definition of T(V), this property holds iff there exists a leaf node z such that $\forall i \; i \in \Theta \; z(i) = \omega$.

Theorem 2.5 Given a vector addition system R(V), it is decidable whether R(V) is finite.

This result follows directly from theorem 2.4.

Theorem 2.6 Given two vector addition systems V and V′, $R(V) \subseteq R(V')$ is undecidable.

Rabin's proof for this result appears in Baker [34].

It should be noted that the general reachability problem i.e., given $x \in \mathcal{N}^r$ is $x \in R(V)$, is not determined by the reachability tree. However, an algorithm for solving the reachability problem has been found [35]. Therefore, the reachability problem is decidable (although the computational complexity is not known).

This completes our study of vector addition systems. We next demonstrate the correspondence between vector addition systems and marked petri nets. Let $M = \langle P,T,I,O,\mu \rangle$ be an arbitrary marked petri net. The corresponding vector addition system V(d,W) may then be constructed in the following manner:

let the dimension $r = |P|$

let $d = \mu_o$ where μ_o is an initial marking for M

let $W = \{w_k \mid t_i \in T, p_j \in P \; w_k(j) = O(p_j,t_i) - I(p_j,t_i)\}$

Note that $|W| = |T|$ and that elements of W reflect the net change in marking resulting from the firing of a transition

in M. This leads to the following theorem relating the reachability sets of M and V:

Theorem 2.7 $Q(M) = R(V)$

Proof. i). $Q(M) \supseteq R(V)$. Let $x \in R(M)$. Then $\mu_0 \stackrel{t_1}{\Rightarrow} \mu_1 \stackrel{t_2}{\Rightarrow} \mu_2 \stackrel{t_3}{\Rightarrow} \ldots \stackrel{t_n}{\Rightarrow} x$. By definition of w_k, $F(\mu, t_i) = \mu + w_k$ and substituting for t_1, t_2, t_3, ..., t_n in the firing sequence $\mu + w_1 + w_2 + \ldots w_n = x$. Since $\mu_0 = d$ by definition, $x \in R(V)$.

ii). $Q(M) \subseteq R(V)$. Let $x \in R(V)$. Then $x = d + w_1 + w_2 + \ldots w_n$. By the same reasoning as case 1, $\mu_0 \stackrel{t_1}{\Rightarrow} \mu_1 \stackrel{t_2}{\Rightarrow} \ldots \stackrel{t_n}{\Rightarrow} x$.

Using theorem 2.7 and the results for vector addition systems we state the following:

Theorem 2.8 Given a marked petri net M with initial marking μ_0, it is decidable if the reachability set Q(M) is finite.

Theorem 2.9 Given a marked petri net M with initial marking μ_0, it is decidable if there exists $k \in \mathcal{N}$, such that for all elements of the reachability set Q(M), each entry $\mu(i) \leq k$.

Definition 2.9 A marked petri net M is <u>k-bounded</u> iff $\exists k \mid \forall \mu \; \mu \in Q(M) \Rightarrow \mu(i) \leq k$.

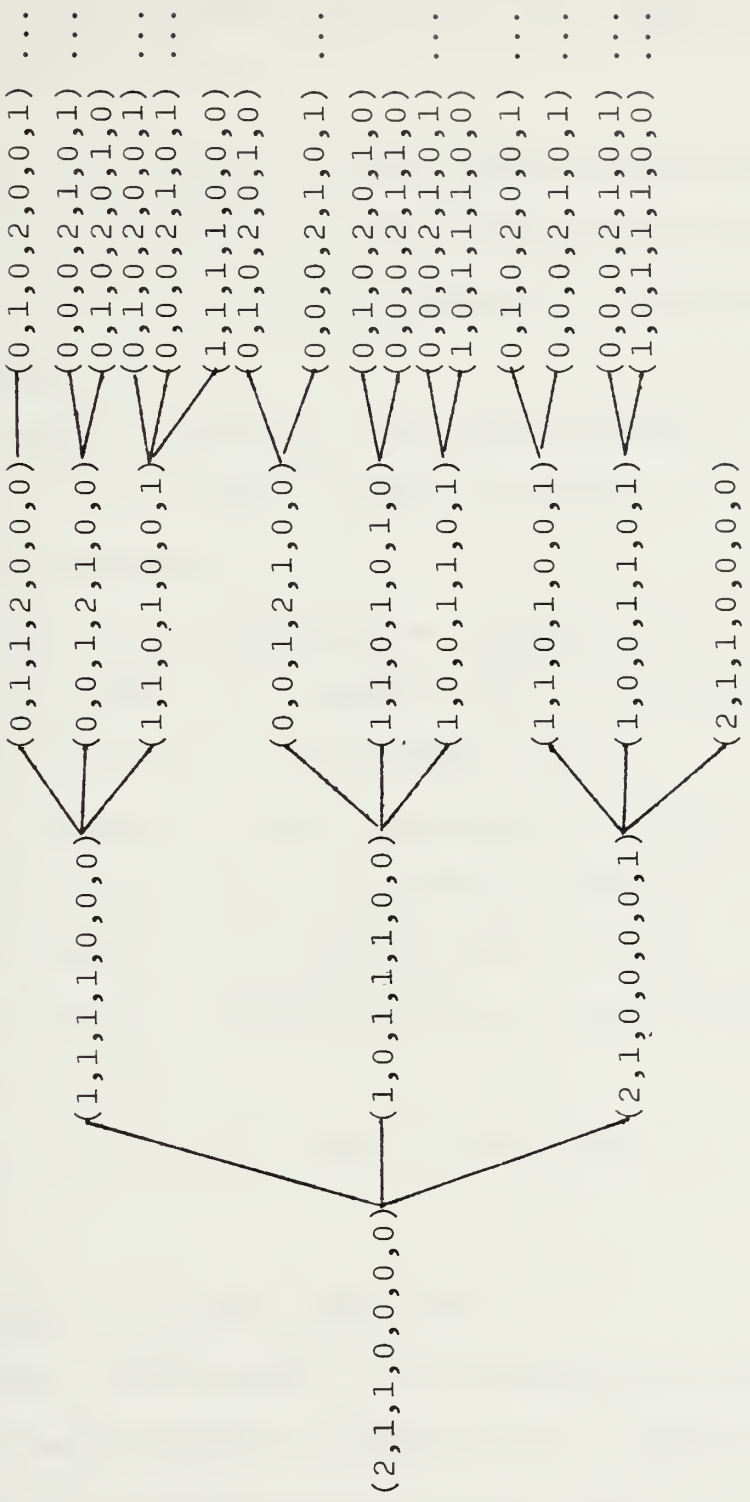


Figure 2.5. Reachability tree for Example 1.

Definition 2.10 A marked petri net M is <u>safe</u> iff it is k-bounded with k = 1 .

We complete this section by once again considering Example 1. Define the corresponding vector addition system:

V = (d,W) with d = [2,1,1,0,0,0,0] as indicated by Figure 2.2.

W = { [-1,0,0,1,0,0,0] , [-1,-1,0,1,1,0,0] ,
[0,1,-1,0,-1,1,0] , [0,0,-1,0,0,0,1] ,
[0,0,1,0,0,0,-1] }

Next, construct the corresponding reachability tree T(V). The tree is shown in Figure 2.5. By inspection, the reachability set R(V) is determined to be:

{ [2,1,1,0,0,0,0] , [1,1,1,1,0,0,0] , [1,0,1,1,1,0,0] ,
[2,1,0,0,0,0,1] , [0,1,1,2,0,0,0] , [0,0,1,2,1,0,0] ,
[1,1,0,1,0,0,1] , [1,1,0,1,0,1,0] , [1,0,0,1,1,0,1] ,
[0,1,0,2,0,0,1] , [0,0,0,2,1,0,1] , [0,1,0,2,0,1,0] ,
[0,0,0,2,1,1,0] }

This (finite) set is also Q(M) and therefore M is k-bounded with k = 2.

B. SUBCLASSES OF MARKED PETRI NETS

The nets investigated in the preceding parts of this section are more properly referred to as generalized petri nets. Analysis of generalized nets has proved to be somewhat intractable. As a result, several properties of petri nets have been studied which define subclasses of the generalized

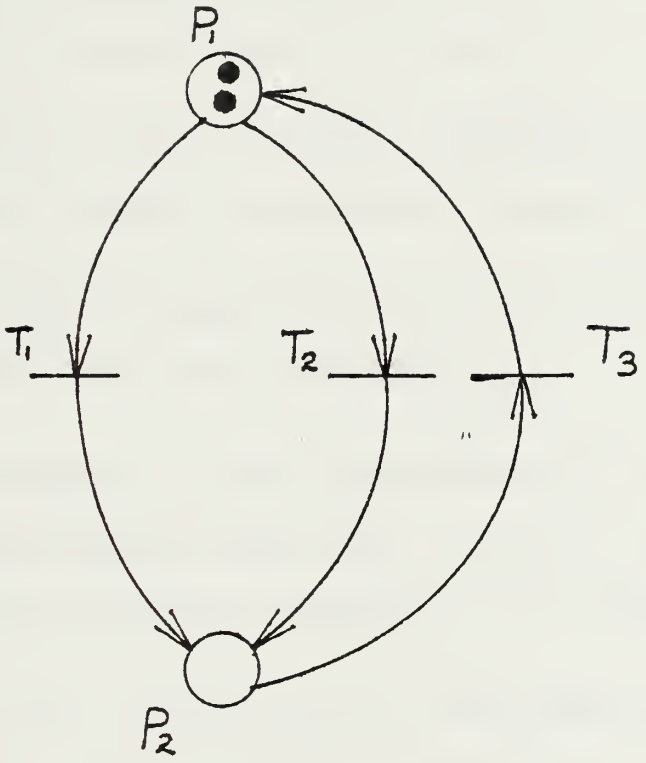


Figure 2.6 Nondeterministic net -- both t1 and t2 are enabled.

nets. These restrictions appear to be justified in that many (if not most) actual systems may be modeled by the restricted nets.

Definition 2.11 A transition $t \in T$ is <u>live</u> for marking μ_0 iff

$$\forall \mu\, \mu_0 \stackrel{\sigma}{=>} \mu \;\Longrightarrow\; \exists \sigma',\ \mu \stackrel{\sigma'}{=>} \mu \wedge t_i \in S_K$$

where S_K is the enabling set for μ_K.

If a transition is live, a firing sequence may always be found which will allow it to be fired indefinitely often.

Definition 2.12 A marked petri net M is live iff

$$\forall t\ t \in T \Longrightarrow t \text{ is live for marking } \mu_0$$

In systems, liveness is often associated with the problem of deadlock. Hack [36] has shown that liveness is equivalent to the reachability problem; therefore, it is decidable.

Definition 2.13 A place $p_i \in P$ is <u>conflict free</u> iff

$$\forall \mu_K\, t_1, t_2, \ldots, t_n \in p_i^{\bullet} \wedge t_i \in S_K \Longrightarrow$$
$$\neg \exists t_j\ j \neq i \wedge t_j \in S_K$$

For any marking, a conflict free place may not enable more than one transition. In Figure 2.6, place p_1 is not conflict free since both t_1 and t_2 are enabled.

Definition 2.14 A petri net M is conflict free iff

$$\forall p\ p \in P \Longrightarrow p \text{ is conflict free.}$$

A stronger statement concerning places is the following:

Definition 2.15 A place p is <u>decision free</u> iff

$$|{\bullet}p_i| = |p_i{\bullet}| = 1 .$$

Definition 2.16 A marked petri net M is a <u>marked graph</u> iff

$\forall p\ p \in P \Longrightarrow$ p is decision free.

A final property of petri nets may be defined:

Definition 2.17 A marking μ_K is <u>persistent</u> iff

$$\forall \mu\ t \in S \wedge (\mu_k \overset{\sigma}{\Rightarrow} \mu_s) \Longrightarrow t \in S_s \vee t \in \sigma.$$

A persistent marking is one in which an enabled transition remains enabled until it is fired.

Definition 2.18 A petri net M is persistent iff

$\forall t\ t \in T \Longrightarrow$ t is persistent for μ_0.

The question of persistence is important in the analysis of petri nets and therefore we present a decidability theorem for these nets:

Theorem 2.10 Given a k-bounded, marked petri net M, it is decidable whether M is persistent.

Proof. Since M is k-bounded, its reachability set Q(M) is finite. For each $\mu_K \in Q$ construct the enabling set S_k. For each $t_i \in S_K$ determine S_{k+1} where $\mu_k \overset{t_i}{\Rightarrow} \mu_{k+1}$. If $S_{k+1} \supseteq S_K - \{t_i\}$, M is persistent.

Note that all conflict free nets (and therefore all marked graphs) are persistent. We next consider some aspects of the petri net classes having these properties.

1. Marked Graphs

Marked graphs have been extensively studied by Commoner and Holt, among others [11,12]. Here we present some pertinent results of their work.

Recall from graph theory the following three definitions:

Definition 2.19 Let $D = \langle A,R \rangle$ be a digraph with nodes a and b. A directed path from a to b is a finite sequence of nodes $P = (c_0, c_1, \ldots, c_n)$ such that $c_0 = a$, $c_n = b$, and for all c_i with $0 \leq i \leq n$ $c_i R c_{i+1}$. If $a = b$, P is a directed circuit.

Definition 2.20 A digraph $D = \langle A,R \rangle$ is strongly connected if for every two nodes $a,b \in A$, there is a directed path from a to b and b to a. If there is an undirected path from a to b, D is connected.

Definition 2.21 A component of a digraph D is a connected subdigraph of D which is not a proper subdigraph of any connected subdigraph of D.

The constraint on the input and output transitions to a place makes it possible to replace every place with a single directed arc between its input and output transitions. In

this way the petri net reduces to a digraph simplifying analysis. The marked graph may then be more simply defined as:

Definition 2.22 A marked graph $M' = \langle T', E', \mu' \rangle$ is a three-tuple where

$T' = T$

$E' : T \times T \rightarrow \{\emptyset, 1\}$

$$\text{where } E(t_i, t_j) = \begin{cases} 1 \text{ if } \exists p \; O(p, t_i) = 1 \\ \quad \wedge I(p, t_j) = 1 \\ \emptyset \quad \text{otherwise} \end{cases}$$

$\mu' : E' \rightarrow \mathcal{N}$ where

$\mu'(e_{ij}) = \mu(p_m)$

where $e_{ij} \in E \wedge {}^{\bullet}p_m = \{t_i\} \wedge p_m{}^{\bullet} = \{t_j\}$

(Here we have synonymously defined the edge set $E' = \{e_{ij}\}$ where the edge directed from node i to node j $e_{ij} \in E'$ iff $E'(t_i, t_j) = 1$

Additionally define the token count N as:

Definition 2.23 The token count of a graph is a function

$N : P \rightarrow \mathcal{N}$ where

if $P \subseteq E$ then $N(P) = \sum_{i=1}^{n} \mu(e_i)$ for $e_i \in P$

Figure 2.7 shows the graphical representation of a marked graph where the number of tokens on an edge e_{ij} corresponds to $\mu'(e_{ij})$. The following theorems and proofs appeared in [11]:

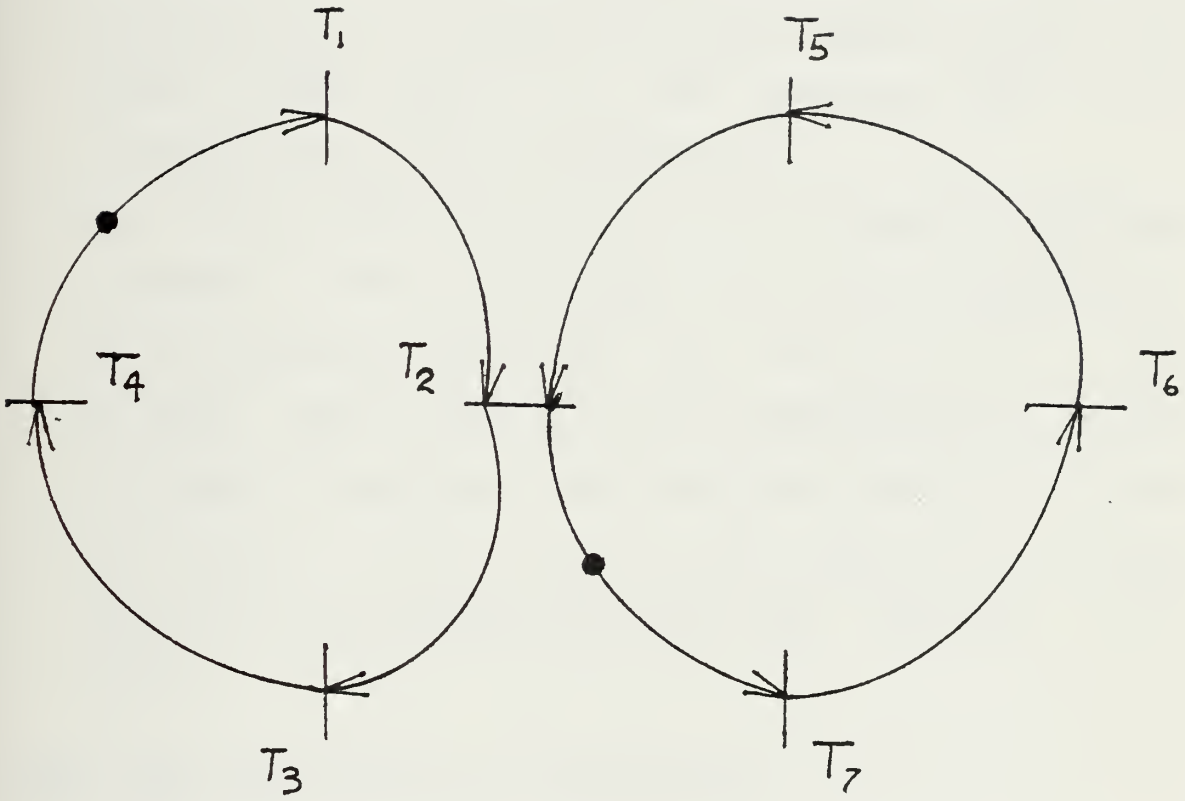


Figure 2.7 Marked graph. Note that tokens are placed on the arcs.

Theorem 2.11 A marking μ' is live iff the token count $N > \emptyset$ for every directed circuit in M.

Proof. i). Assume μ' is live and $N(B) = \emptyset$ for some directed circuit B and e_{ij} is an edge in that circuit. By definition, a sequence σ exists which enables t_i. After firing $\mu'(e_{ij}) = 1$ and therefore $N(B) = 1$ in contradiction with the assumption.

ii). Assume $N(B) > \emptyset$ and t_j is a node in directed circuit B. If t_j is enabled, t_j is live. Otherwise let t_i be a node in B such that e_{ij} is in B. If t_i is enabled, fire it resulting in t_j being enabled. If not, continue to backtrack. Since the path length of B must be finite in the directed circuit beginning and ending with t_j, this procedure must halt and therefore t_j is live.

This leads immediately to a corollary:

Corollary 2.11.1 A marking which is live remains live after firing.

Theorem 2.12 A live marking is safe iff every edge in the graph is in a directed circuit with token count $N(P) = 1$.

Proof. Clearly, the token count of any directed circuit is constant. If $N(P) = 1$ then, the edges of the directed

circuit must be safe. If $N(P)\,K > 1$, by the same process as in theorem 2.11 transitions in the circuit may be fired until k tokens appear on the same edge.

Theorem 2.13 For every finite, strongly connected graph G there exists a live and safe marking for the corresponding marked graph M′.

Proof. By definition, each edge in G must lie in a directed circuit. Since G is finite, a finite number of directed circuits exist. Therefore construct μ' by placing one token arbitrarily on each directed circuit. The conditions of theorems 2.11 and 2.12 are met and M′ is live and safe.

2. State Machine Decomposable Nets.

This class of petri nets has been studied by Ramchandani [8]. A state machine is defined as:

Definition 2.24 A marked petri net M is a <u>state machine</u> iff

$$\forall t \quad t \in T \wedge [I(p_i,t) > \emptyset] \wedge [I(p_j,t) > \emptyset] \Longrightarrow p_i = p_j$$

$$\forall t \quad t \in T \wedge [O(p_i,t) > \emptyset] \wedge [O(p_j,t) > \emptyset] \Longrightarrow p_i = p_j$$

That is, $\forall t \; |{}^\bullet t| = |t^\bullet| = 1$.

This restriction results in nets which are functional equivalents of finite state machines, hence the name state

machine. Note that state machines allow conflict i.e., nondeterminism.

Definition 2.25 A subnet M_i is a strongly connected component of M.

Definition 2.26 A petri net M is state machine decomposable iff

$$\exists\{M_i\} \mid \bigcup_i P_i = P \wedge \bigcup_j T_j = T \wedge \forall M_i \; M \text{ is a state machine.}$$

Several properties of state machines may easily be shown.

Theorem 2.14 The token count N(M) is constant for a marked state machine.

Proof. Assume $N(M) = C$ and $t_i \in T$. By definition $|{\bullet t_i}| = |t_i{\bullet}| = 1$. Let $\{p_1\} = {\bullet t_i}$ and $\{p_2\} = t_i{\bullet}$ with $N(p_1) = c_1$ and $N(p_2) = c_2$. If t_i does not fire, the token count of p_1 and p_2 are unaffected by t_i. If t_i does fire, by definition $N(p_1)$ $c_1 - 1$ and $N(p_2) = c_2 + 1$. Summing we have:

$$\sum_{p_1 p_2} N(p) = c_1 - 1 + c_2 + 1 = c_1 + c_2$$

and the token count does not change.

Corrolary 2.14.1 Every marked state machine is bounded.

Proof. Since the token count N(M) is constant, the maximum number of tokens at any place is N(M); therefore M is bounded.

3. Consistent Petri Nets.

An additional class of petri nets are those for which there exist consistent current assignments.

Definition 2.27 A petri net M is <u>consistent</u> iff there exists a function $\Phi: T \rightarrow I$ such that

1. $\forall t_i \in T \quad \Phi(t_i) = \Phi_i$
2. $\forall p \in P \quad \sum_{i=1}^{n} \Phi_i \times I(p,t_i) = \sum_{j=1}^{n} \Phi \times O(p,t_j)$

   where in the summation $n = |T|$.

The function Φ is analogous to an electromagnetic flux and the Φ_i's are referred to as the currents associated with transitions t_i. Note that part 2 of the definition is an expression of Kirkhoff's Law. Part 2 requires solving the following set of linear equations:

$$\begin{vmatrix} C_{I\,1,1} & \cdots & C_{I\,1,n} \\ \vdots & & \vdots \\ C_{I\,m,1} & \cdots & C_{I\,m,n} \end{vmatrix} \cdot \begin{vmatrix} \Phi_1 \\ \vdots \\ \Phi_n \end{vmatrix} = \begin{vmatrix} C_{O\,1,1} & \cdots & C_{O\,1,n} \\ \vdots & & \vdots \\ C_{O\,n,1} & \cdots & C_{O\,n,n} \end{vmatrix} \cdot \begin{vmatrix} \Phi_1 \\ \vdots \\ \Phi_n \end{vmatrix}$$

where C_I and C_O are the incidence matrices for M.

If a non-zero solution exists, M is consistent.

Example 3. Consider the net in Figure 2.8.

Construct $C_I = \begin{vmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{vmatrix}$ $\qquad C_O = \begin{vmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{vmatrix}$

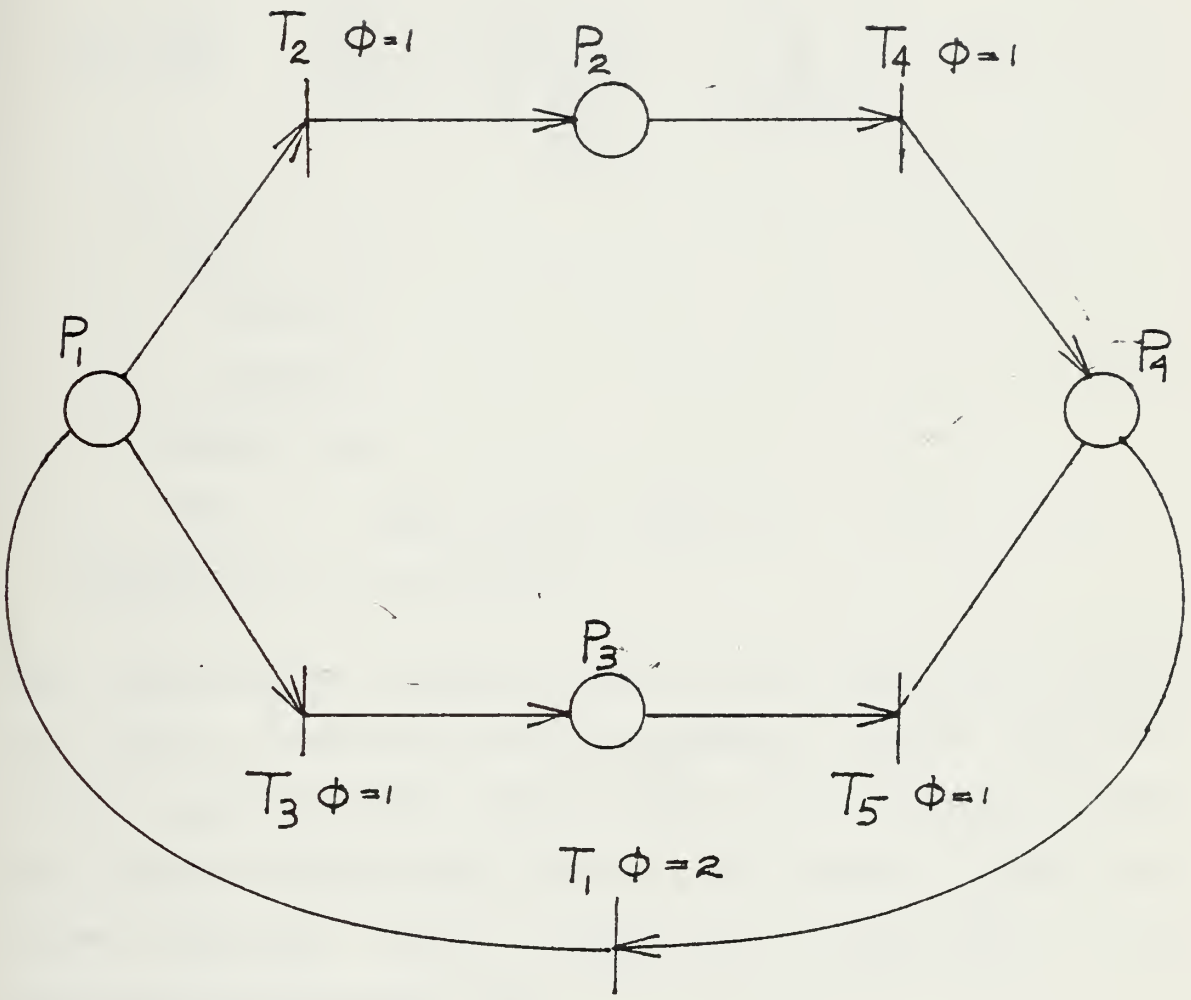


Figure 2.8 Consistent Petri net

$$C_I \times \Phi = \begin{vmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{vmatrix} \bullet \begin{vmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \end{vmatrix} = \begin{vmatrix} \Phi_2 + \Phi_3 \\ \Phi_4 \\ \Phi_5 \\ \Phi_1 \end{vmatrix}$$

$$C_O \times \Phi = \begin{vmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{vmatrix} \bullet \begin{vmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \\ \Phi_5 \end{vmatrix} = \begin{vmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 + \Phi_5 \end{vmatrix}$$

$$\Phi_2 + \Phi_3 = \Phi_1$$

$$\Phi_2 = \Phi_4$$

$$\Phi_3 = \Phi_5$$

$$\Phi_4 + \Phi_5 = \Phi_1$$

$$\Phi_1 = 2 \qquad \Phi_2 = \Phi_3 = \Phi_4 = \Phi_5 = 1$$

Therefore M is consistent.

The significance of consistency is that a consistent system will cycle -- given an initial marking μ_o, there exists a firing sequence such that $\mu_o \overset{\sigma}{\Rightarrow} \mu_o$. An inconsistent system will either consume tokens and halt or produce tokens and become unbounded. These results are summarized in two theorems by Ramchandani [8].

Theorem 2.15 A petri net M is consistent iff

$\exists \mu_o, \sigma$ such that $\mu_o \overset{\sigma}{\Rightarrow} \mu_o$ is a cycle.

Proof. i). Let $\Phi_1, \Phi_2 \ldots \Phi_n$ be the consistent currents of the transitions of M. Let $\mu_o(p_i) = \Phi + \Phi + \Phi = C$ where $\Phi_1 \ldots \Phi_n$ correspond to $t_i \ldots t_n \in p_i \bullet$. Then let

$\sigma = \{t_1, t_2 \ldots\}$ where the multiplicity of t_i in σ is equal to its current giving $\mu_o \overset{\sigma}{\Rightarrow} \mu_j$. Since the sum of the currents into and out of a place is 0, $\mu_o(p_i) = \mu_j(p_i)$ and $\mu_o = \mu_j$. Therefore σ is a cycle.

ii). Let $\mu_o \overset{\sigma}{\Rightarrow} \mu_o$ be a cycle. Then $\mu_j(p_i) \Rightarrow \mu_o(p_i)$. Let $k_1, k_2, \ldots, k_n$ be the multiplicity of transitions $t_1, t_2, \ldots, t_n \in {}^\bullet p_i$ and $l_1, l_2, \ldots, l_n$ be the multiplicity of transitions $t_1, t_2, \ldots, t_n \in p_i^\bullet$. $\sum_{j=1}^{n} k_j = \sum_{j=1}^{n} l_j$. Then let $\Phi(t_j) = k_j$ if $t_j \in {}^\bullet p_i$ and $\Phi(t_j) = l_j$ if $t_j \in p_i^\bullet$. This is a consistent current assignment and therefore M is consistent.

Theorem 2.16 A petri net M with a live, bounded marking is consistent.

Proof. Since M is bounded, its reachability graph Q(M) is finite. Since M is live, there exists a strongly connected subgraph which contains μ. Therefore, there exists a directed circuit in the subgraph firing all transitions. This is a cycle and therefore M is consistent.

We conclude our description of petri net subclasses by considering the hierarchical relationships between classes. In the sections on marked graphs and state machines, it has been shown that both are contained within the class of bounded petri nets. It is easy to show that the containment

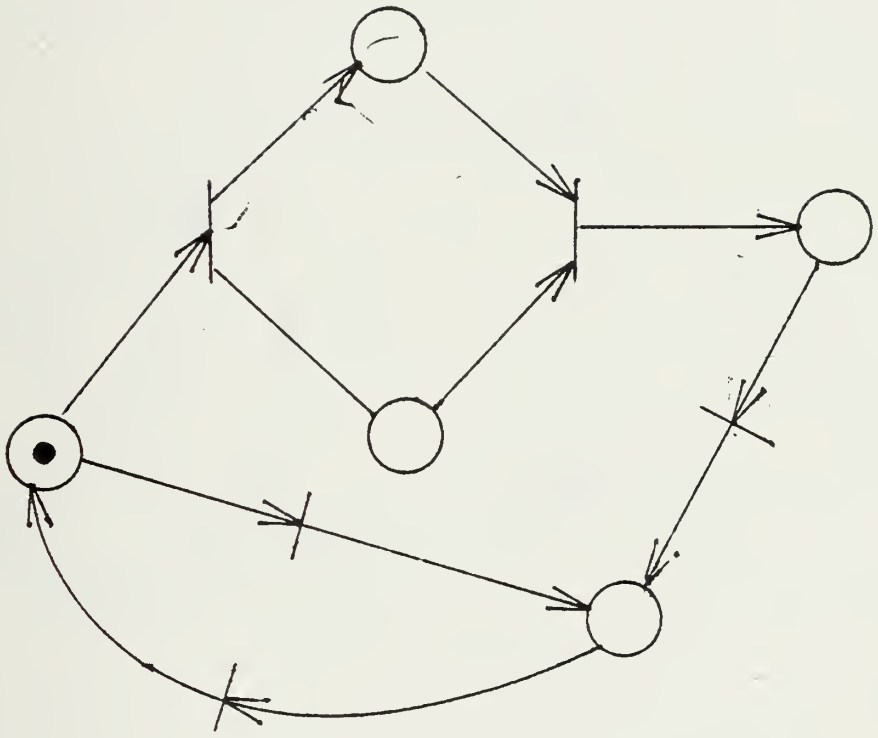


Figure 2.9 Bounded Petri net

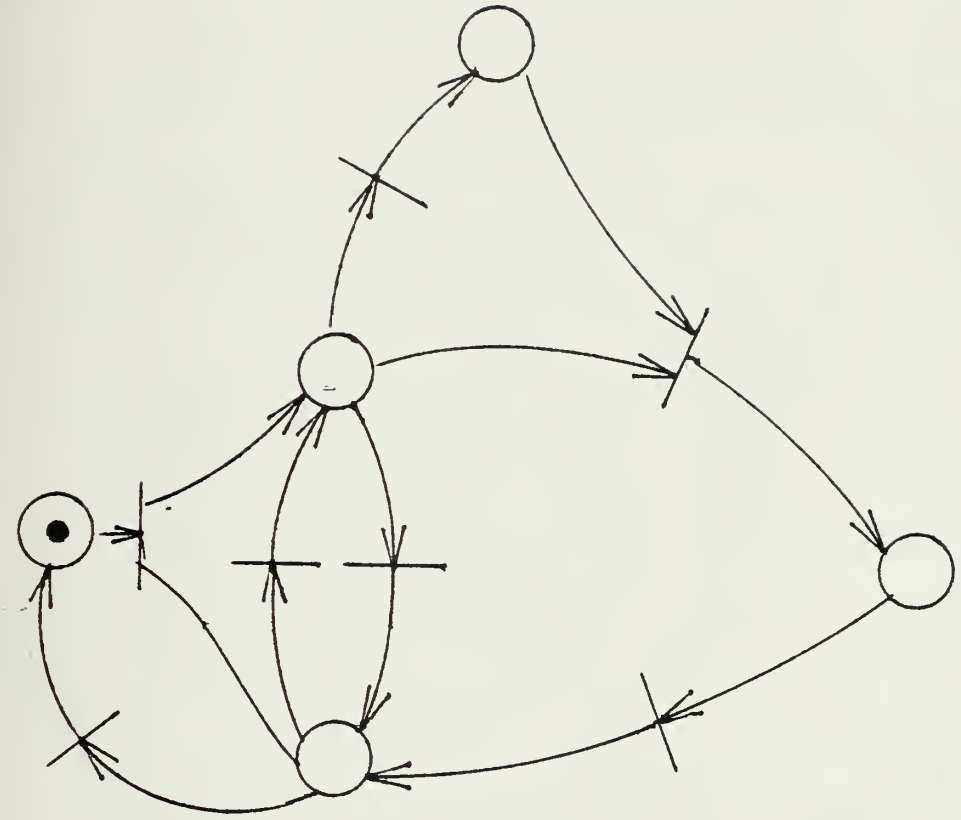


Figure 2.10 Unbounded, non-live, consistent net

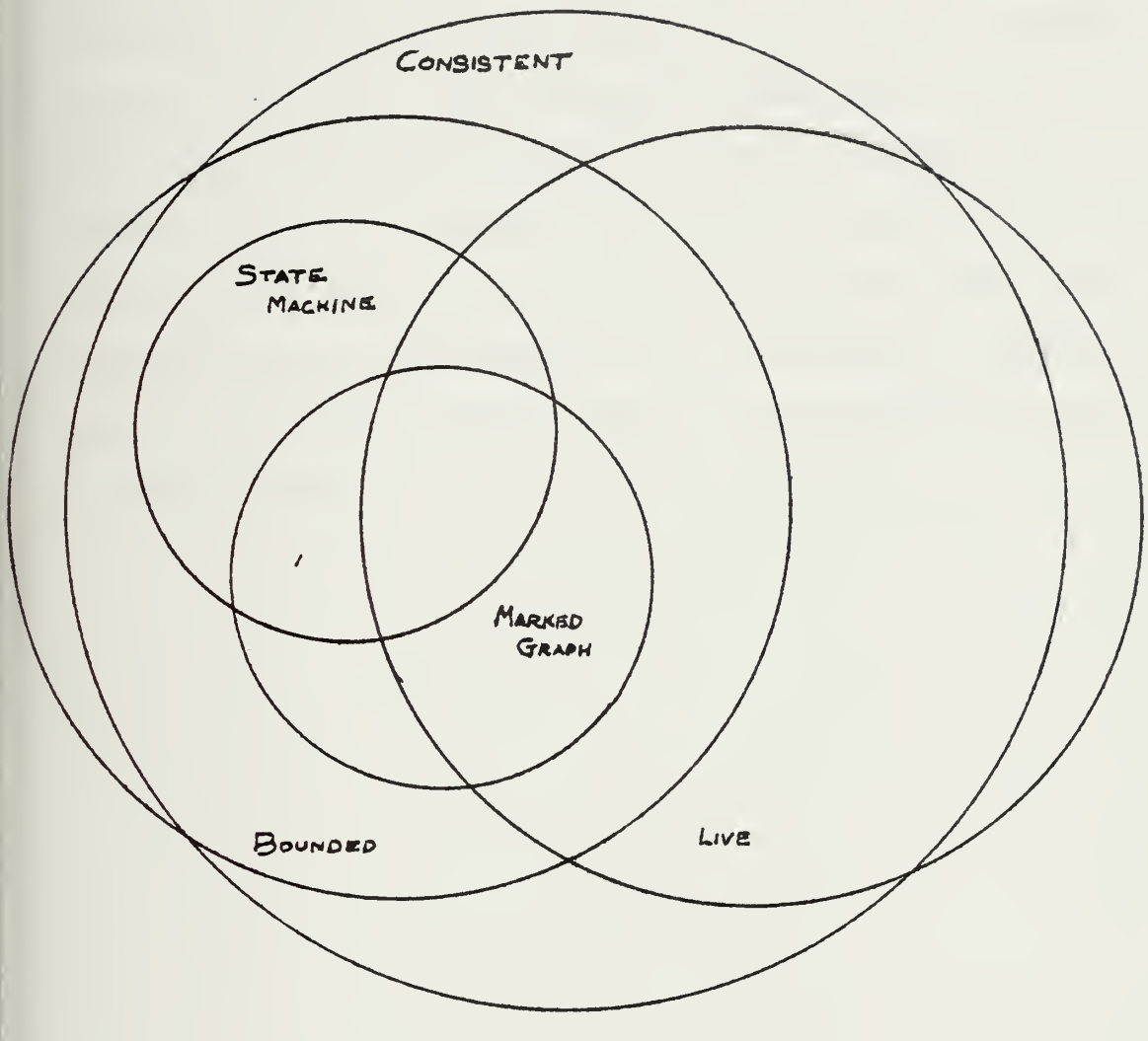


Figure 2.11 Petri net hierarchy

is proper. Figure 2.9 shows a net which is bounded but neither a marked graph nor a state machine. The intersection of the class of marked graphs and state machines are a degenerate class we call sequential processes. Theorem 2.16 showed that all live, bounded marked nets are consistent. Once again, the containment is proper -- Figure 2.10 is an example of an unbounded non-live net which is consistent. Finally note that all persistent or conflict free nets are deterministic and therefore may be reduced to decision free nets i.e., marked graphs. These relationships are summarized in Figure 2.11.

III. STOCHASTIC PETRI NETS

To this point we have analyzed petri nets on the basis of their structural characteristics. To utilize petri nets for computing performance measures, it is necessary to introduce the concepts of time and nondeterminism to the basic model. We turn first to the question of modeling time in petri nets.

A. TIMED EVENTS IN PETRI NETS

An important concept in Section II was the marking, or system state, of the net. The marking gives an instantaneous description of the token content of each place in the net. The marking was changed as a result of the firing of a transition. We defined a firing sequence as an allowable ordering of transition firings in accordance with the rules for enabling transitions. By controlling the dynamics of the transition firing process, it is possible to analyze the changes in system state as a function of time.

Several authors have addressed the question of adding timing considerations to petri nets. Two different interpretations have resulted from this work. Sifakis [37] has proposed that once enabled, transitions fire instantaneously. A delay is then introduced before tokens are available at the output places for possible enablings of other transitions. An alternative view is that taken by Ramchandani [8] and Zuberek [38]. In their models, once

enabled, transitions fire after a delay called the firing time. After the firing time is completed, the net changes state by moving the appropriate tokens. These two interpretations appear to be equivalent. We choose to use the latter interpretation to conform to the usual notion of a queueing service center.

We first consider the case where transition firings occur at discrete time epochs $\tau_1, \tau_2, \ldots, \tau_n, \ldots$ where τ_n is the instant of the nth firing.

Definition 3.1 The <u>system state</u> of a marked petri net M is a function:

$$U: T \rightarrow \mu$$

where $T = \{\tau_1, \tau_2, \ldots, \tau_n, \ldots\}$,

$U(\tau_0) = \mu_0$,

and $U(\tau_n) = \mu_K \Rightarrow \exists\, j \mid U(\tau_{n-1}) = \mu_j \wedge \mu_j \overset{t}{\Rightarrow} \mu_K$

U is a step function with discontinuities at those instants of time in which the system changes state.

Definition 3.2 The <u>firing time</u> X of a petri net is a function:

$$X: T \rightarrow R^+$$

where $\forall t_i \in T \quad X(t_i) = \tau_i = x$

By this definition, x_i is the time required for a firing of the transition t_i.

By specifying the firing time function X, it is possible to describe $U(\tau_n)$. In this section we consider the case where the transition firing times x_i are constant for all transitions. This restriction makes it possible to describe a total ordering of transition firings for persistent nets. In the case of allowed conflict between transitions, it is necessary to impose a priority scheme on transition firings to resolve the conflict (that is, to make the firing deterministic). This ordering amounts to a restriction on the firing sequences which are allowed. Those sequences σ for which the ordering holds are termined feasible firing schedules. Ramchandani has shown that for timed marked graphs and live, safe, and persistent petri nets a periodic feasible firing schedule exists. He additionally derived an upper bound on the computation rate (cycle period) for state machine decomposable nets. The bound is given by:

$$\rho^{max} = \min[\rho_1, \rho_2, \rho_n]$$

where ρ_n is the cycle time for each circuit in the corresponding net and is given by:

$$\rho_n = \frac{N(C_i)}{\sum_{j=1}^{n} x_j} \cdot \Phi_i$$

where $N(C_i)$ is the token count for circuit C_i, x_j is the sum of the firing times for the transitions in C_i, and Φ_i is the current associated with that circuit in a minimal current assignment. Ramchandani also argues that this formula can serve as a first order approximation where the

mean firing times $\overline{x_i}$ are substituted for the deterministic firing time x_i in the formula. A slightly different derivation of the same result was made by Ramamoorthy and Ho [39].

B. MARKOV ANALYSIS OF PETRI NETS

Research into the behavior of timed petri nets to date has concentrated on deterministic nets with constant firing times. As noted in Section I, this approach fails to account for the randomness and uncertainty which characterize actual systems. We propose that nondeterminism be modeled probabilistically in the net. Our method differs from previous work by emphasizing the stochastic nature of the system state (marking). In this way, it is possible to apply the known methods of queueing theory to analyze the probabilistic properties of these nets which we call stochastic petri nets. We introduce in this section two sources of nondeterminism which will be modeled by the stochastic petri nets -- random transition firing times and the probabilistic firing of simultaneously enabled transitions.

We first show that if the firing time x for each transition is considered to be a random variable rather than a deterministic one, it is possible to analyze the token content of the net as a stochastic process.

Definition 3.3 The firing time X_i is an independent, identically distributed, random variable such that $\forall t_i \in T$ $x_{i,n}$ is the firing time for the nth firing of transition t_i.

The requirement that X_i is independent and identically distributed is necessary for our derivation of a Markov chain representation for the net state space. In the case of computer networks, Kleinrock [40] has investigated this requirement which he has named the message independence condition. This assumption is somewhat artificial in that it implies, for example, that the time required to process the same message at two nodes is independent. This difficulty notwithstanding, results for an analysis of the ARPA net show some evidence for the validity of this assumption [41]. Applying the methods of probability theory, the firing time distribution may be defined as:

Definition 3.4 For all $t_i \in T$, if X_i is the firing time for t_i,

$S_i(x) = P[X_i \leq x]$

In the usual way we define the density function:

Definition 3.5 For all transitions $t_i \in T$, the firing time density function is given by

$s_i(x) = d/dx\ [S_i(x)] = p[X_i = x]$

Likewise define the moments:

Definition 3.6 The nth moment of the firing time density function is given by:

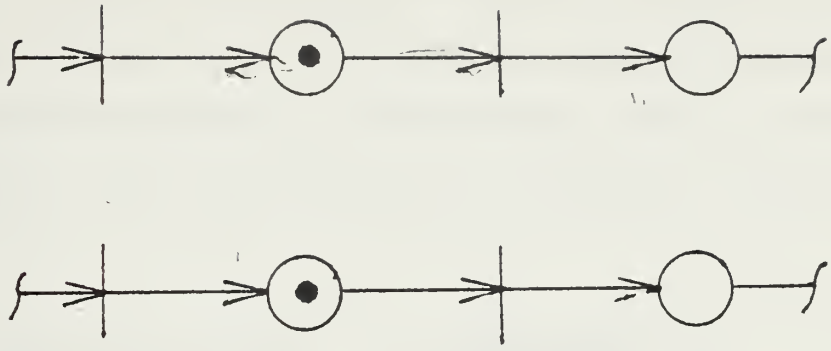
$$E[X_i]^n = \int_{-\infty}^{\infty} x^n s_i(x)dx$$

and in particular $E[X_i] = \overline{X_i} = 1/\mu_i$ is the mean firing (service) time*.

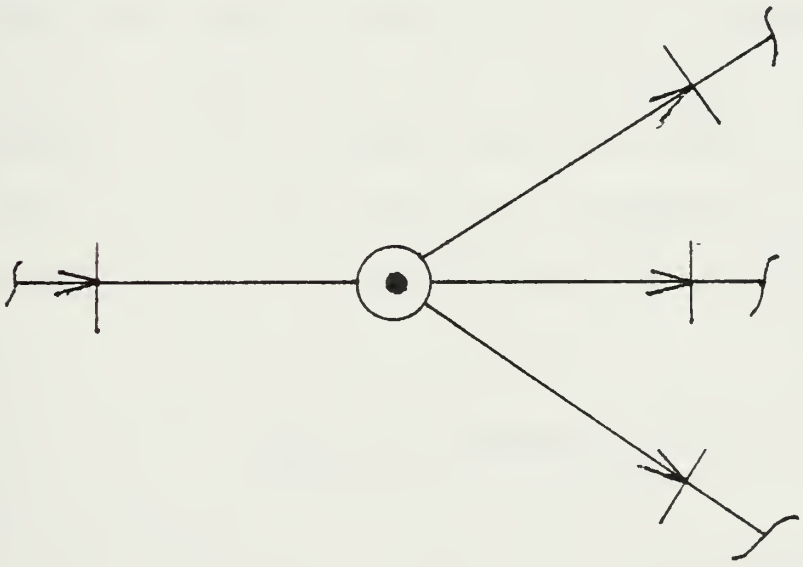
These definitions make it possible to model systems by applying the appropriate distributions to the firing time function. It is clear that the feasible firing sequences σ for a petri net are no longer deterministic. It is necessary to consider now this nondeterminism can occur in the net. Referring to Figure 3.1(a), it can be seen that transitions t_2 and t_4 are both enabled. With random firing times, the sequences t_2, t_4 and t_4, t_2 are both possible. In this instance the effect of the nondeterminism is unimportant to overall system operation since the two processes are independent at this point. A more interesting situation is depicted in Figure 3.1(b). The transitions t_2, t_3, and t_4 are in conflict. To analyze this type of nondeterminism it is necessary to specify the branching probabilities for each of the possible paths. There are several possible ways this may be accomplished. For example, each simultaneously

---

* The mean service rate μ is an unfortunate conflict in notation. Due to its long standing use in the literature, we will use this notation pointing out its meaning where confusion might exist with the earlier definition of marking.



(a)



(b)

Figure 3.1 Nondeterminism in petri nets. (a) independent processes. (b) transitions with conflict.

enabled transition may begin firing at once with the transition which completes firing first causing the marking to change. The problem of determining which transitions are simultaneously enabled is a significant one. We can simplify the problem by restricting our analysis to free choice places.

Definition 3.7 A place p is <u>free choice</u> iff

$$|p_i\bullet| = 1 \vee \forall t_j \in T \quad t_j \in p_i\bullet \Longrightarrow \bullet t_j = \{p_i\}$$

A free choice place is one which either has a unique output transition or is the only input place to each of its output transitions. This restriction ensures that a marking for p will either enable all of its output transitions simultaneously, or will enable none of them. With this restriction, it is now possible to define the branching probabilities for the output transitions of a free choice place.

Definition 3.8 The <u>branching probability</u> for a free choice place p_i with $p_i\bullet = \{t_1, t_2, \ldots, t_n\}$ is

$b_{ij} = P[t_j \text{ will fire} \mid t_1, t_2, \ldots, t_j \text{ are enabled}]$

such that:

$$\emptyset < b_{ij} \leq 1$$

and

$$\sum_{j=1}^{n} b_{ij} = 1 \text{ where } n = |p_i\bullet|$$

We make the assumption that these probabilities are functions of the firing distributions for the output transitions only and in particular, that they are constant and independent of the marking.

By allowing these sources of nondeterminism, the system state U may be expressed as a stocnastic process.

Definition 3.9 Let $U(\tau)$, the system state of a petri net M, be a random variable and a function of time where $U(\tau) = [u_1(\tau), u_2(\tau), \ldots, u_n(\tau)]$ such that $u_i(\tau) = \mu_\tau(p_i)$. $U(\ )$ is a discrete state, continuous time stochastic process described by the probability distribution:

$$f_\mu(\mu;\tau) = p[U(\tau_1) = \mu_1, U(\tau_2) = \mu_2, \ldots, U(\tau_n) = \mu_n]$$

$$\mu_1, \mu_2, \ldots, \mu_n \in Q(M).$$

$U(\tau)$ describes the manner in which the system moves between states in the reacnability set. It should be noted that the distribution $f_\mu(\mu;\tau)$ is equivalent to expressing the probability that some feasible firing sequence σ exists such that $\mu_0 \stackrel{\sigma}{\Rightarrow} \mu$. Therefore, it is possible to extend the definition of liveness.

Definition 3.10 A transition $t \in T$ is live for state U iff

$$\forall U \ \exists \sigma_1 \ U_j \stackrel{\sigma_1}{\Rightarrow} U \wedge p[\sigma_1] > 0 \Longrightarrow$$

$$\exists \sigma_2 \ U \stackrel{\sigma_2}{\Rightarrow} U_K \wedge p[\sigma_2] > 0 \wedge t_i \in S_K$$

Theorem 3.1 For a live stochastic petri net M with marking and initial marking $\mu_o \in Q(M)$,

$p[U(\tau) = \mu_i] > \emptyset$

Proof. Assume $p[U(\tau) = \mu_i] = \emptyset$. Then the probability that a firing sequence σ exists such that $\mu_o \overset{\sigma}{\Rightarrow} \mu_i$ is zero in contradiction with the assumptions of Definition 3.10 and therefore M cannot be live.

We are interested in determining the conditions for which U is a Markov chain, that is, when

$$p[U(\tau_{n+1}) = \mu_{n+1} \mid U(\tau_n) = \mu_n, U(\tau_{n-1}) = \mu_{n-1}, \ldots, U(\tau_0) = \mu_o] = p[U(\tau_{n+1}) = \mu_{n+1} \mid U(\tau_n) = \mu_n].$$

In addition, we are interested in determining the stationary (time independent) state probabilities if they exist. We first consider the case of state machine decomposable petri nets.

C. CLOSED PETRI NET SYSTEMS

To express the system state transitions as a Markov process it is first necessary to derive expressions for the arrival and departure processes for the places in the net. This may be accomplished by considering the net as a collection of nodes, each of which has a well-defined behavior, and in particular, for which the arrival and departure processes may be expressed analytically.

A node in a petri net is defined as:

Definition 3.11 A node n in a petri net is a subset of the set $\{P \bigcup T\}$ such that

$\forall t_i \in T \quad t_i \in n \iff$

$\forall p_j \quad p_j \in {}^\bullet t_i \iff p_j \in n$

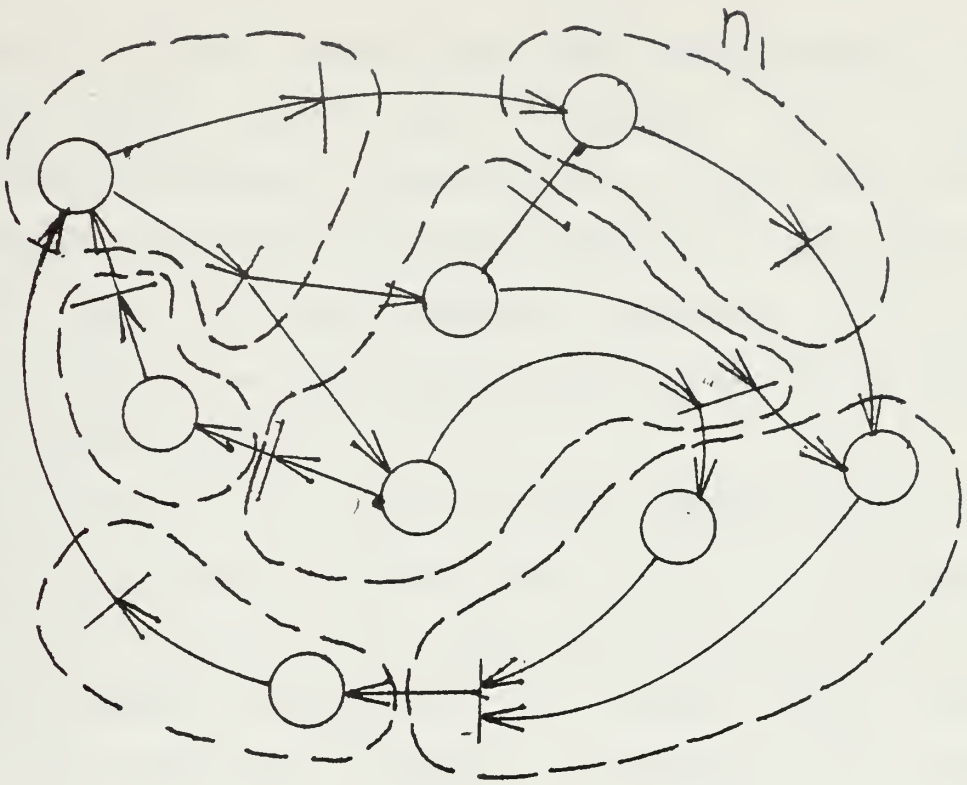
$\wedge \forall t_k \quad t_k \in p_j{}^\bullet \iff t_k \in n$

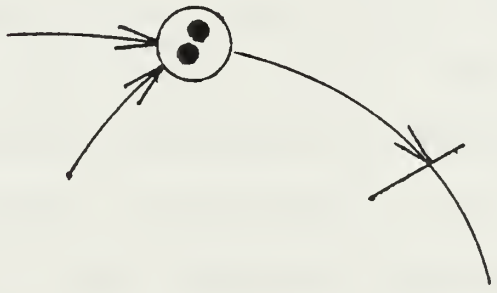
Theorem 3.2 The node set $[n] = n_1, n_2, \ldots, n_m$ is a partitioning of a petri net M.

Proof. i.) $\bigcup_{i=1}^{m} n_i = \{P \bigcup T\}$. By definition, every transition t is trivially an element of some node n_k. Assume there exists a place p_j such that $p_j \notin (n_1, n_2, \ldots, n_m)$. By the definition of a petri net, p_j has (at least) one input transition t_i. Since t_i is an element of some node, by Definition 3.11 p_j is an element of that node.

ii.) $\forall n_a, n_b \in [n] \quad n_a = n_b \vee n_a \bigcap n_b = \phi$. Assume there exists some transition $t_i \in \{n_a \bigcap n_b\}$. Let p_j be an input place to t_i and an element of n_a. Then by Definition 3.11 p_j is also an element of n_b. Now let t_k be an output transition to p_j and an element of n_a. Again by definition t_k is also an element of n_b and therefore $n_a = n_b$.

Likewise, assume there exists some place $p_j \in \{n_a \bigcap n_b\}$. By the same reasoning $n_a = n_b$. Therefore, the set of nodes defines a partitioning of M. QED.





(b)

Figure 3.2 Queueing nodes in petri nets. (a) partitioning of an arbitrary net. (b) single place/transition node.

Figure 3.2(a) shows the node partitioning of an arbitrary petri net. Now consider the single place/transition node n_i. Figure 3.2(b) is this node with an arbitrary marking. Since we have assumed that the transition in this node is firing whenever enabled, we have the important result that this node is equivalent to a single server queueing system.

The marking for p, which we have defined as the state element u, includes the tokens being fired or waiting to fire. Two features of this representation should be noted. First, all tokens are identical; no token classes exist. Second, no queueing discipline is modeled in the system. This places a restriction on the ability to derive analytic solutions for the system.

To describe the operation of the node, that is, determine the local state probabilities $P[U(\tau) = u]$, it is necessary to characterize the arrival and departure processes. It is well known that the exponential distribution is the only continuous distribution for which the Markov property holds. In addition, it has been shown [33] that arbitrary work conserving queueing disciplines result in equilibrium product form state probability solutions for exponential firings. Therefore we assume the firing time distribution S(x) to be:

Definition 3.12 The firing time distribution for transition t_i is given by

$$S_i(x) = 1 - \exp(-\mu_i x)$$

where the mean firing rate is μ_i.

The requirement that the queueing discipline be work conserving means that no knowledge of the firing time requirements is used in selecting tokens for firing. It is clear from the indeterminate nature of the tokens in petri nets that this is indeed the case and therefore we are justified in asserting that the Markov property holds.

Finally, it is necessary to determine the arrival process for the node. In the case of closed networks (and the petri nets we have considered thus far), the arrivals are made up of departures from other nodes in the net. It is assumed that upon completion of firing, the tokens immediately enter their associated output places. Therefore, the arrival process may be characterized as:

Definition 3.13 The arrival rate Γ_i for a place p_i is given by:

$$\Gamma_i = \sum_{j=1}^{m} \Gamma_j \, b_{ji}$$

where Γ_j is the arrival rate for node n_j, b_{ji} is the brancning probability that a departure from node n_j arrives at node n_i, and m is the rank of [n].

Note that we have not specified the arrival distribution itself. It has been shown by Burke's theorem [42] and several extensions (for example [3]), that in many cases the arrival process is asymptotically Poisson.

To summarize, we have made the following assumptions concerning the petri net state transition process:

1. Firing times are independent.

2. Firing times are continuous and exponentially distributed.

3. The queueing discipline is work conserving.

4. Nondeterministic transitions between nodes are determined by constant branching probabilities.

5. There is no overhead in transition firings; transitions fire whenever enabled.

In Section II we considered the class of state machine decomposable nets and state machines. Figure 3.3 is an example of such a net with its associated nodes.

Theorem 3.3 Each node in a state machine contains exactly one place.

Proof. By definition, each transition has a single input place. Trivially then, there must be at least one place in each node. Without loss of generality, assume there exists a node with two places p_1 and p_2 with $t_i \in p_1\bullet$. By implication, $t_i \notin p_2\bullet$ and therefore $p_2 \notin \bullet t_i$ in

contradiction with the definition of a node. Therefore, there is at most one place in each node. QED.

The next result follows immediately:

Corollary 3.3.1 The places in a state machine are free-choice.

Since the places are free-choice, it is possible to assign the branching probabilities to the output transitions. These transitions are not multiple servers; rather, they represent the possibility for tokens which must be handled differently. It is necessary therefore, to create composite places to deal with this requirement. The resulting net we call the stochastic equivalent petri net (SEN).

Definition 3.14 The <u>stochastic equivalent net</u> for a petri net M is constructed as follows:

For every place p in P, assign a set $\{\pi_1, \pi_2, \ldots, \pi_n\}$ where $n = |p_i\bullet|$, $\forall t_k \in T \quad I(\pi_j, t_k) = I(p_i, t_j)$, and if $p_i\bullet = \{t_1, t_2, \ldots, t_n\}$

$$O(\pi_j, t_k) = \begin{cases} O(p_i, t_j) & j=k \\ \emptyset & \text{otherwise} \end{cases}$$

In the associated graph, each edge from a transition in $\bullet p_i$ to the set $\{\pi_1, \pi_2, \ldots, \pi_n\}$ is joined by an arc and labeled with the appropriate branching probability.

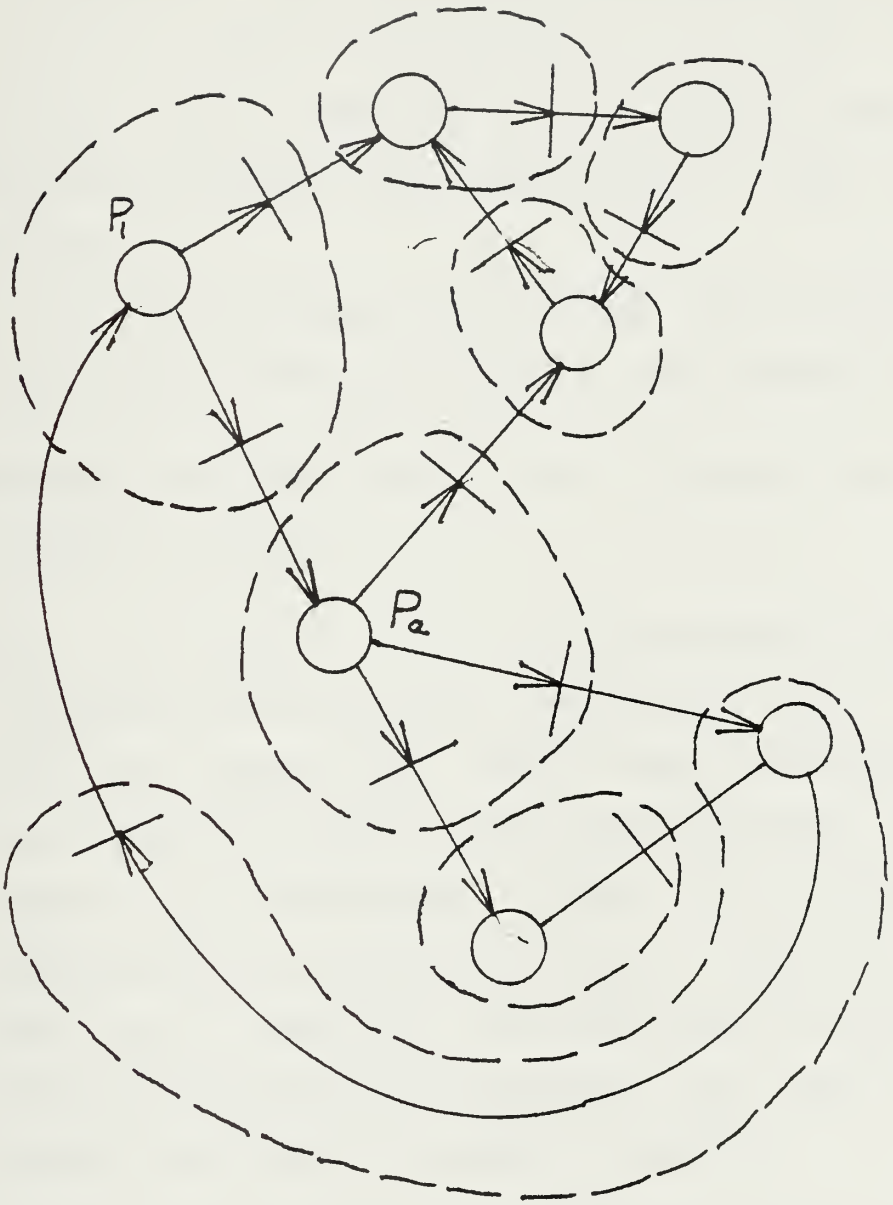


Figure 3.3 Partitioning of a petri net into a node set.

The SEN for the petri net in Figure 3.3 is depicted in Figure 3.4. Each node in the SEN consists of a single place and transition, with the branching probabilities occurring at the output from the transitions. It is clear from the definition that the SEN has the same properties of boundedness, liveness, and consistency as the associated petri net.

It is now possible to obtain the analytic solution for the petri net by treating is as a closed queueing network.

Theorem 3.4 The SEN for a state machine with a live marking is ergodic.

Proof. By theorem 2.12, a state machine is bounded and therefore the state space (reachability set) is finite. Lien has shown [43, thm 11] that the state space for this class is strongly connected. Therefore, the state space is irreducible. Since it is finite, the probability of reaching some state, $P[U_i]$, is greater than zero. Therefore the state space is recurrent and non-null. By definition therefore, the state space is ergodic and likewise the SEN is ergodic.

Thus, equilibrium state probabilities exist for the system; that is,

$$P[U] = \lim_{\tau \to \infty} P[U, \tau]$$

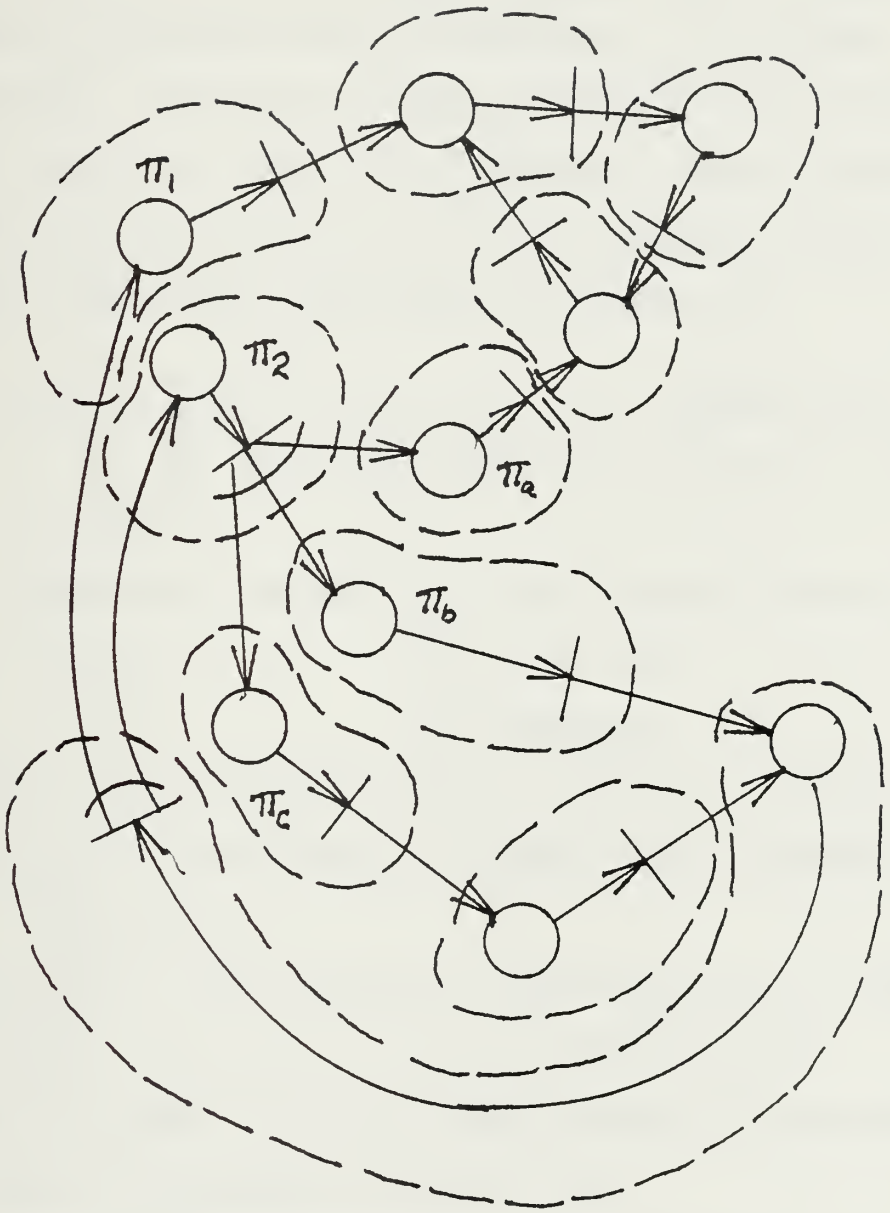


Figure 3.4 Stochastic equivalent net for the petri net in Figure 3.3.

This type of system was solved by Gordon and Newell [30]. At equilibrium, the derivative $d/d\tau(P[U,\tau])$ must vanish for each state in the state space. This allows the global balance equations for the system to be written. For some state U_i, the rate of change of probability is determined by the rate of flow of probability into the state due to arrivals and the rate of flow of probability from the state due to departures. This may be written as:

$$d/d\tau(P[U,\tau)] = P[(u_1,u_2,\ldots,u_n)] \sum_{i=1}^{n} \delta(u_i)\cdot\mu_i - \sum_{i=1}^{n}\sum_{j=1}^{n} \delta(u_i)\mu_i b_{ij}\cdot P[(u_1,u_2,\ldots,u_j-1,\ldots,u_i+1,\ldots,u_n)] = \emptyset$$

where $\delta(u_i)$ is the unit step function given by

$$\delta(u_i) = \begin{cases} \emptyset \text{ if } u_i = \emptyset \\ 1 \text{ otherwise} \end{cases}$$

These equations may be solved directly to within a constant which may be determined by adding the requirement that

$$\sum_{i=1}^{n} P[U_i] = 1$$

The product form solution to these equations is [30]

$$P[U] = P[(u_1,u_2,\ldots,u_n)] = \{1/G(K)\} \prod_{i=1}^{n} x_i^{u_i}$$

where $K = N(U)$ and the x_i are solutions to equations

$$\mu_i\cdot x_i = \sum_{j=1}^{n} \mu_i\cdot x_j\cdot b_{ji} \quad (i=1,2,\ldots,n).$$

The normalization constant G(K) is given by

$$G(K) = \sum_{U} \prod_{i=1}^{n} x_i^{u_i}$$

Algorithms have been found for computing G(K) and the x_i [42].

D. OPEN PETRI NET SYSTEMS

Since the SEN for a state machine has been shown to be equivalent to closed networks of queues, the question arises as to whether petri nets can be defined which are analogous to open networks. Such a system may model the occurrence of external events such as the arrival of interrupts. Alternatively, the model may represent a communications system where messages enter and are removed from the system at various points. To incorporate these external events, Definition 2.1 may be extended as follows:

Definition 3.15 An open marked petri net OM is a marked petri net where

$$\forall t_i \in T \qquad {}^{\bullet}t_i = \phi \iff t_i \text{ is a source}$$
$$t_i{}^{\bullet} = \phi \iff t_i \text{ is a sink}$$

It is assumed that these transitions may source or sink an infinite number of tokens. The firing rates μ_i are defined as before except that $\gamma_i = \mu_i$ is the mean arrival rate for source t_i.

The possibility of external arrivals requires a modification of the earlier arrival process definition.

Definition 3.16 The <u>arrival rate</u> Γ_i for a node is given by

$$\Gamma_i = \gamma_i + \sum_{j=1}^{n} \Gamma_j b_{ji}$$

It can be seen that Definition 3.13 is a special case where $\gamma_i = 0$ for $i = 1,2,\ldots,n$.

Several observations can be made concerning the ways in which the properties of liveness, boundedness, and consistency are affected by the addition of sources and sinks. First, consider the case of marked graphs. Recall that for a marked graph, each place is conflict free. Figure 3.5(a) shows part of a marked graph. To meet the conflict free requirement, source and sink transitions can be added to the net only at existing transitions (with intervening places being added). In Figure 3.5(b), a source transition st1 has been added at transition t_2 and in Figure 3.5(c), a sink transition st2 has been added at transition t_4.

Theorem 3.5 Liveness in marked graphs is unaffected by the addition of source or sink transitions.

Proof. By definition, a source transition is live, and therefore liveness is unaffected by the addition of a source. Now consider a transition in the net t_i with a sink transition/place pair added to the output. If t_i is live, it can be fired by some sequence which will then enable the sink transition; therefore, the sink transition is live and liveness is conserved.

Theorem 3.6 A marked graph remains bounded after addition of source and sink transitions except in the places associated with those transitions.

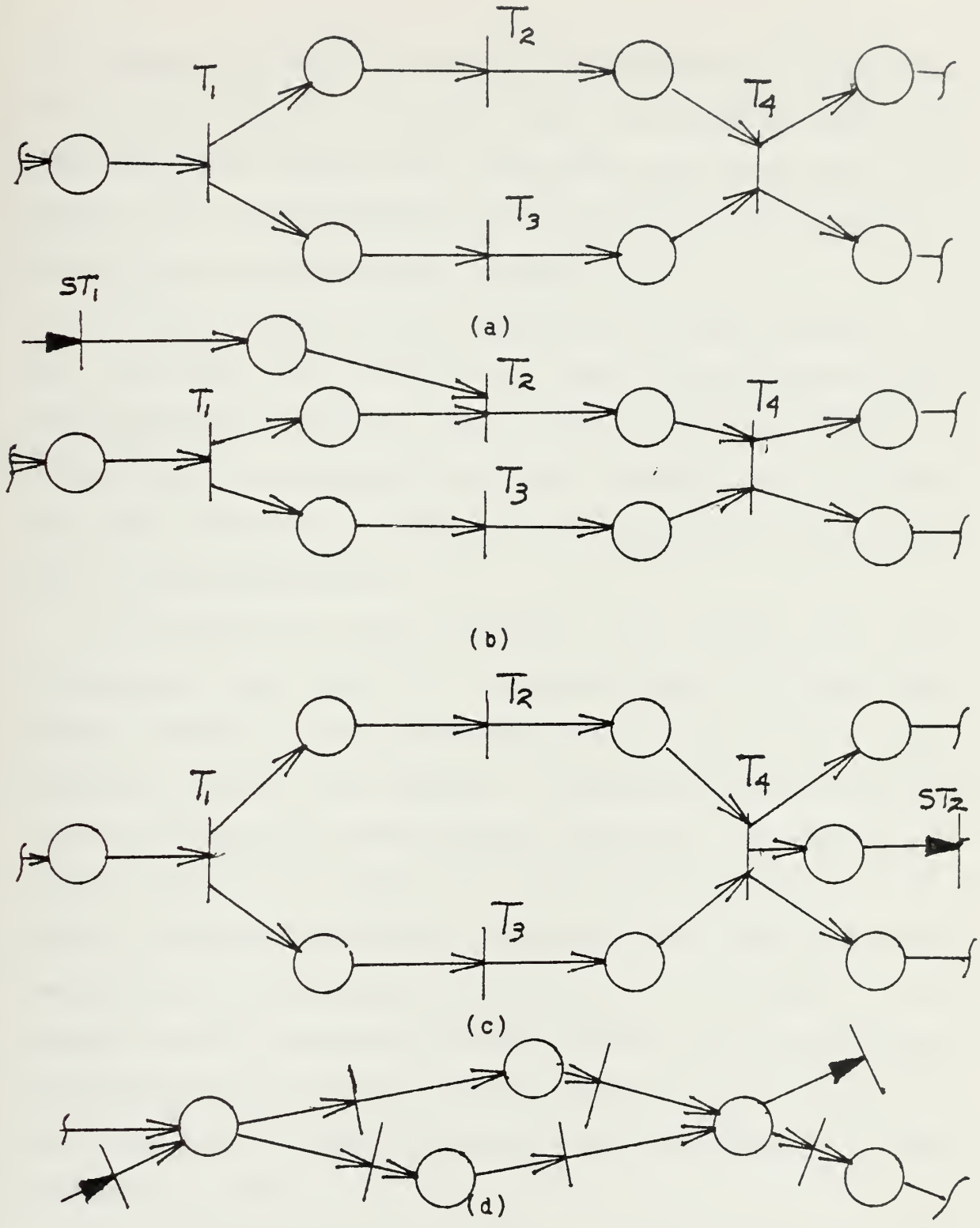


Figure 3.5 Addition of source and sink transitions to petri nets. (a) marked graph. (b) marked graph with source st1. (c) marked graph with sink st2. (d) state machine net with sources and sinks.

This result is clearly true by the definition of transition firing. It should be noted that the general results of queueing theory require that the places associated with the source and sink transitions be unbounded (if exponential arrivals and departures are assumed).

In terms of the dynamic operation of marked graphs, it can be seen from Figure 3.5 that the existence of sink transitions has no effect. The source transitions operate by controlling the enabling of the net transitions to which they are connected; they set an upper bound on the firing rate of the transition.

In the case of state machines, the source and sink transitions are added to existing places in the net (see Figure 3.5(d)). The nondeterministic nature of state machines results in changes to the properties of the net after the addition of the sources and sinks. For example, if a sink transition is added to a live state machine, the net could eventually terminate since the sink must eventually become enabled resulting in the loss of a token to the system. When sources are added, the net will become become unbounded. Since the source can be fired arbitrarily often, the token count of the output place to the source can become arbitrarily large.

By generalizing the closed network queueing model developed in the previous section, the solution for state machines with sources and sinks may be obtained. Once again,

the net is transformed into its stochastic equivalent. A sink is added to a node j by adjusting the branching probabilities of the transitions in j so that the probability of departure from the system at node j, P[d], obeys:

$$P[d] = 1 - \sum_{i=1}^{n} b_{ji} \text{ where n is } |t_i^*|.$$

A source is added to node j by inserting an edge directed into the place in j and labeled with the mean arrival rate.

Since it is possible for the net to become unbounded after adding sources and sinks, it is necessary to require that $\Gamma_j < \mu_j$ for every node in the SEN to ensure that the net remains ergodic.

Once again, the solution involves writing the global balance equations for the system, i.e.:

$$\forall U_j \quad \sum_{i=1}^{n} P[U_i] \text{ (rate of flow from } U_i \text{ into } U_j)$$
$$= P[U_j] \text{ (rate of flow out of } U_j)$$

Basket [32] has shown that the general product form solution for this system is of the form

$$P[U] = Cd(U)f_1(u_1)f_2(u_2)\ldots f_n(u_n)$$

where C is the normalization constant needed to ensure

$$\sum_{i=1}^{n} P[U_i] = 1 ,$$

d(U) is an expression for the exogenous arrival rate,

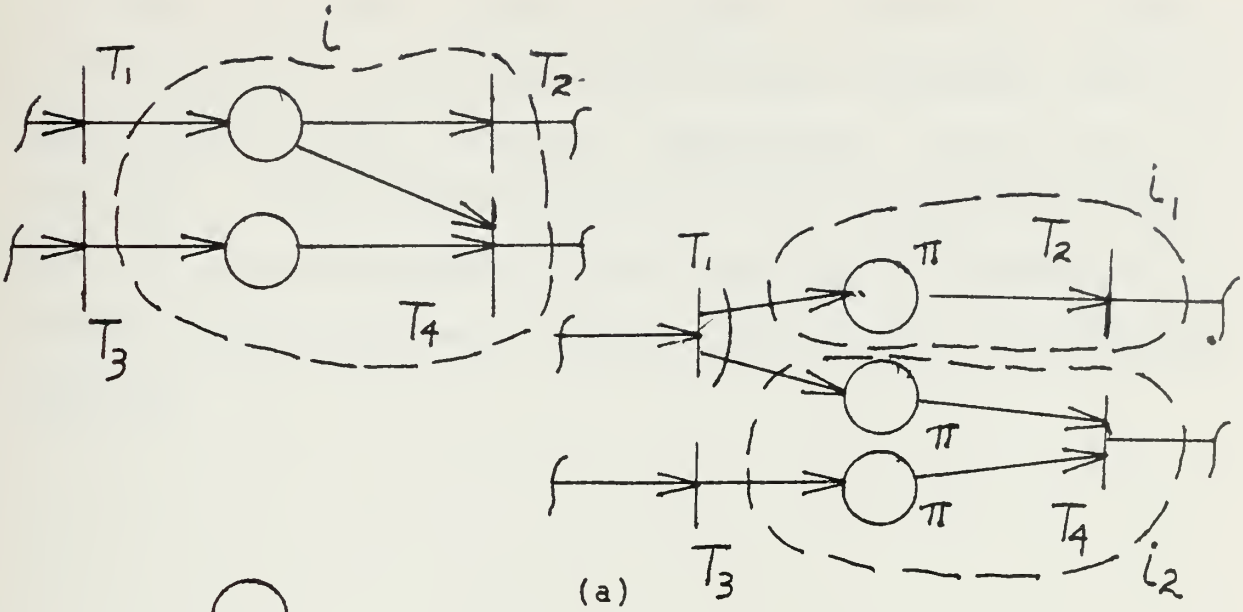
$d(U) = \sum_{i=1}^{n} \gamma_i$ for Poisson arrivals at constant rate γ_i,

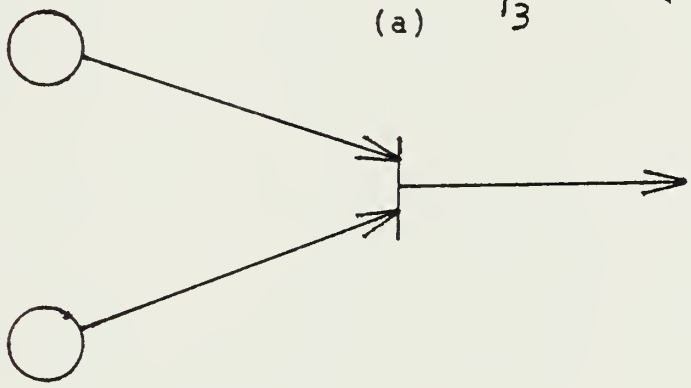
and $f_i(u_i)$ is a function of the queue discipline.
For FCFS, $f_i(u_i) = 1/\mu_i \prod_{j=1}^{u_i} \Gamma_j = \left(\frac{\Gamma_i}{\mu_i}\right)^{u_i}$

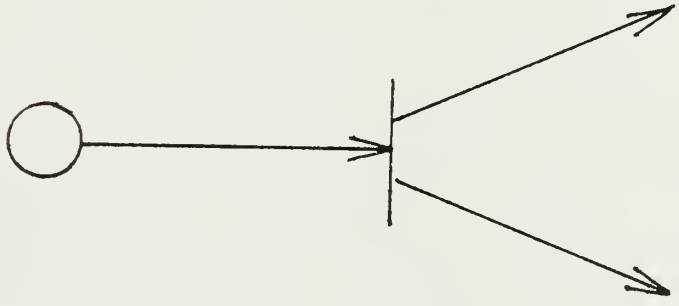
E. CONSISTENT PETRI NET SYSTEMS

The most general petri net class we consider is that of live, bounded, and consistent nets. The solution to this class proceeds identically to that used earlier. The net is first transformed into its stochastic equivalent net form by the method of Definition 3.13. Figure 3.6(a) shows how an arbitrary node in the petri net is transformed. Note that in the resulting node i2, two places are required as input to transition t_4 and therefore this node does not model the simple queue/server pair which was seen earlier for state machines. Another possibility for a SEN node is that the transition has multiple outputs with branching probabilities b_{ij} all equal to 1. These two situations are shown in Figure 3.6(b). It has been shown [44] that these types of nodes (also referred to as join and fork nodes) do not satisfy local balance and therefore product form solutions do not exist for this class of nets. However, since the net is bounded, by the results of Chapter II the reachability set (and state space) is finite and therefore the global state equations can be solved numerically [45].

In the case of consistent nets with sources and sinks, it is possible for the net to become unbounded since the sources may fire arbitrarily often and therefore finding the







(b)

Figure 3.6 Transformation of petri nets with forks and joins into SEN. (a) Arbitrary node transform. (b) join and fork nodes.

solution to the global balance equations becomes intractable. In this case, approximation techniques such as aggregation or the diffusion approximation [46] must be resorted to. The applicability of these techniques to petri nets is an open question, but one which has the potential to extend the modeling power of petri nets to more interesting systems.

IV. CONCLUSIONS

This thesis has addressed the problem of computer system performance analysis through the use of the petri net model. The model has wide applicability to the analysis of both hardware and software systems, particularly those which exhibit concurrency or asynchronous operation. Due to the power of the petri net approach, it is necessary to restrict the structure of the nets resulting in a hierarchical class relationship between petri net types.

The classes of greatest interest in system modeling are marked graphs, state machines, and consistent nets. All three of these classes were shown to have the properties of boundedness, liveness, and consistency which are useful in the verification of computer system correctness. Of the three classes, the class of consistent nets is the most useful in modeling in that it can represent the greatest range of possible systems -- and is also the most difficult to analyze mathematically.

It was seen that the general problem of reachability, and the set of states in which a marked petri net could enter, was a primary consideration in the analysis of the nets. In particular, an algorithm was presented for determining the state space for the class of bounded petri nets.

By defining the firing times for the transitions in the net, it was possible to extend the analysis of petri nets to their dynamic execution. Random firing times allowed the nets to model data dependent events. This concept led to the identification of nondeterminism in the petri net execution.

Petri nets with random firing times were shown to be analogous to closed queueing networks. A major difficulty in this approach is the inability to model queueing disciplines in the places. However, if exponential firing is assumed, the analysis can be conducted without detailed knowledge of the structure of the queue. In this context, the stochastic equivalent net was introduced as a method for demonstrating the correspondence between petri nets and queueing networks. This permitted the state probabilities to be determined through the known techniques of queueing theory.

The definition of petri nets was extended to allow for events which take place external to the system itself. Again, the resulting stochastic equivalent net corresponds to a queueing network -- in this case the open network model.

It appears that this approach to petri net modeling can utilize more of the recent results of queueing network theory. For example, solutions have been found for state dependent routing (branching) probabilities, and state dependent arrival and departure rates. Additional work is required to determine if the use of petri net based

stochastic models simplifies the problem of reducing system design criteria and parameters to a form which permits the application of queueing network techniques.

APPENDIX A

LIST OF NOTATION

b_{ij}	probability that transition j will fire given that place i enables transitions $t_1, t_2, \ldots, t_n$
C_I	input incidence matrix for a petri net
C_O	output incidence matrix for a petri net
$E(t_i, t_j)$	the set of edges in a marked graph between transitions i and j
$\Phi(t)$	a current assignment for a transition in a consistent net
$F(U,t)$	the firing function for a transition t and state U in a petri net
Γ_i	the total mean arrival rate at node i
$I(p,t)$	the input function for place p and transition t
M	a marked petri net $\langle P,T,I,O,\mu \rangle$
M′	a marked graph $\langle T',E',\mu' \rangle$
$\mu(p)$	the marking function for a marked petri net (section 2)
μ_i	the mean firing rate for transition i (sec 3)
N	a petri net $\langle P,T,I,O \rangle$
N(P)	the token count for a set of places P
O(p,t)	the output function for place p and transition t
P	the set of places in a petri net
π	a composite place in a stochastic equivalent net

$Q(M)$	the reacnability set (space state) for petri net M
$R(V)$	the reacnability set for a vector addition system V
ρ	the computation rate for a discrete time petri net
σ	a firing sequence $(t_1, t_2, \ldots, t_n)$
S_K	the enabling set $\{t_1, t_2, \ldots, t_n\}$ enabled by marking μ_k
$S(x)$	the transition firing time distribution
$s(x)$	the transition firing time density fuction
T	the set of transitions in a petri net
U_i	the state of a petri net $(u_1, u_2, \ldots, u_n)$ with marking μ_i
V	a vector addition system $\langle d, W \rangle$
W	the vector set in a vector addition system

LIST OF REFERENCES

1. Cox, L. A., Performance Prediction From a Computer Hardware Description, Naval Postgraduate School Report NPS52-79-001, December 1979

2. Anderson, G. A. and Jenson, E. D., Computer Interconnection Structures: Taxonomy, Characteristics, and Examples, Computing Surveys, Vol 7 No 4, December 1975

3. Kobayashi, H. and Konheim, A. G., Queueing Models for Computer Communications System Analysis, IEEE Trans Communications, Vol Com-25 No 1, January 1977

4. Ferrari, D., Computer Systems Performance Evaluation, Prentice-Hall, 1978

5. Petri, C. A., Kommunikation mit Automaten, Schriften des Rheinisch- Westfalischen Institutes fur Instrumentelle Mathematik an der Universitat Bonn, 1962

6. Noe, J. D. and Nutt, G. J., Macro E-Nets for Representation of Parallel Systems, IEEE Trans Computers, Vol C-22 No 8, August 1973

7. Agerwala, T. and Flynn, M., Comments on Capabilities, Limitations and 'Correctness' of Petri Nets, Proc First Annual Symp on Comp Arch, ACM, 1973

8. Ramchandani, C., Analysis of Asynchronous Concurrent Systems by Timed Petri Nets, MIT Project MAC TR-120, PhD Thesis, 1974

9. Peterson, J. L., Computation Sequence Sets, J of Comp and System Sci, Vol 13, pp 1-24, 1976

10. Hack, M., Petri Net Languages, MIT Project MAC CSG Memo 124, 1974

11. Commoner, F. and Holt, A. W., Marked Directed Graphs, J Comp and System Sci, Vol 5 No 7, December 1971

12. Holt, A. W. and Commoner, F., Events and Conditions, Record of Project MAC Conf on Concurrent Systems and Parallel Computation, ACM, June 1970

13. Landweber, L. H. and Robertson, E. L., Properties of Conflict-Free and Persistent Petri Nets, J of the ACM, Vol 25 No 3, July 1978

14. Misunas, D., Petri Nets and Speed Independent Design, Comm of the ACM, Vol 16 No 8, August 1973

15. Patil, S. S. and Dennis, J. B., The Description and Realization of Digital Systems, Digest of Papers, COMPCON 72, IEEE Computer Society, 1972

16. Thornton, J. E., Design of a Computer: The Control Data 6600, Scott, Foresman and Co, 1970

17. Cox, L. A., Performance Prediction of Computer Architectures Operating on Linear Mathematical Models, PhD Thesis, Dept of Comp Sci, University of California--Davis, September 1978

18. Anderson, D. W. and others, The IBM/360 Model 91: Machine Philosophy and Instruction Handling, IBM J Research and Development, Vol 11 No 1, January 1967

19. Stowers, D. M., Computer Architecture Performance Prediction for Naval Fire Control Systems, Masters Thesis, Naval Postgraduate School, December 1979

20. Valette, R., Analysis of Petri Nets by Stepwise Refinements, J of Comp and System Sci, Vol 18, pp 35-46, 1979

21. Merlin, P. M., A Methodology for the Design and Implementation of Communication Protocols, IEEE Trans Communications, Vol Com-24 No 6, June 1976

22. Dijkstra, E. W., Cooperating Sequential Processes, Programing Languages, Genuys, F. ed., Academic Press, 1968

23. Peterson, J. L., Petri Nets, Computing Surveys, Vol 9 No 3, September 1977

24. Karp, R. M. and Miller, R. E., Properties of a Model For Parallel Computations: Determinancy, Termination, Queueing, SIAM J Appl Math, Vol 14, No 6, November 1966

25. Karp, R. M. and Miller, R. E., Parallel Program Schemata, J of Comp and System Sci, Vol 3, pp 147-195, 1969

26. Reiter, R., Scheduling Parallel Computations, J of the ACM, Vol 15 No 4, October 1968

27. Meyer, S. C., An Analytic Approach to Performance Analysis For a Class of Data Flow Processors, Proc 1976 Conf on Parallel Processing, IEEE Computer Society, 1976

28. Miller, R. E., A comparison of Some Theoretical Models of Parallel Computation, IEEE Trans Computers, Vol C-22 No 8, August 1973

29. Jackson, J. R., Networks of Waiting Lines, Operations Research, Vol 5, pp. 518-521, 1957

30. Gordon, W. J. and Newell, G. F., Closed Queueing Systems With Exponential Servers, Operations Research, Vol 15 No 2, March 1967

31. Jackson, J. R., Jobshop-Like Queueing Systems, Management Science, Vol 10, pp. 131-142, 1963

32. Baskett, F. and others, Open, Closed, and Mixed Networks of Queues With Different Classes of Customers, J of the ACM, Vol 22 No 2, April 1975

33. Chandy, K. M. and others, Product Form and Local Balance in Queueing Networks, J of the ACM, Vol 24 No 2, April 1977

34. Baker, H. G., Rabin's Proof of the Undecidability of the Reachability Set Inclusion Problem of Vector Addition Systems, MIT Project MAC CSG Memo 79, July 1973

35. Sacerdote, G. and Tenney, R. L., The Decidability of the Reachability Problem for Vector Addition Systems, submitted for publication

36. Hack, M., The Recursive Equivalence of the Reachability Problem and the Liveness Problem for Petri Nets and Vector Addition Systems, Proc 15th Annual Symp on Switching and Automata, IEEE, 1974

37. Sifakis, J., Use of Petri Nets for Performance Evaluation, Measuring, Modeling and Evaluating Computer Systems, Beilner, H. and Gelenbe, E. eds, North-Holland, 1977

38. Zuberek, W. M., Timed Petri Nets and Preliminary Performance Evaluation, Proc 7th Annual Symp on Comp Arch, IEEE, 1980

39. Ramamoorthy, L. V. and Ho, G. S., Performance Evaluation of Asynchronous Concurrent Systems Using Petri Nets, IEEE Trans Software Engineering, Vol SE-6 No 5, September 1980

40. Kleinrock, L., Communication Nets, McGraw-Hill,1964

41. Kleinrock, L., Performance Models and Measurements of the ARPA Computer Network, Computer Communication Networks, Grimsdale, R. L. and Kuo, F. F. eds., Noordhoff, 1975

42. Buzen, J. P., Computational Algorithms for Closed Queueing Networks with Exponential Servers, Communications of the ACM, Vol 16 No 9, September 1973

43. Lien, Y. E., Termination Properties of Generalized Petri Nets, SIAM J on Computing, Vol 5 No 2, June 1976

44. Towsley, D. F., Local Balance Models of Computer Systems, PhD Thesis, Univ of Texas at Auston, December 1975

45. Stewart, W. J., A Comparison of Numerical Techniques in Markov Modeling, Communications of the ACM, Vol 21, February 1978

46. Chandy, K. M. and Saur, C. H., Approximate Methods for Analyzing Queueing Network Models of Computing Systems, Computing Surveys, Vol 10 No 3, September 1978

INITIAL DISTRIBUTION LIST

		No Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department of Computer Science NPS Code 52 Naval Postgraduate School Monterey, CA 93940	4
4.	Professor L. A. Cox NPS Code 52CL Naval Postgraduate School Monterey, CA 93940	3
5.	LT Scott W. Smart, USN 7409 Edward New Orleans, Louisiana 70126	3
6.	Professor R. Hamming NPS Code 52HG Naval Postgraduate School Monterey, CA 93940	2
7.	Professor J. McGraw Department of Applied Science University of California P.O. Box 808 Livermore, CA 94550	1
8.	Dr. J. Dennis Department of Computer Science Massachusetts Institute of Technology Boston, MA 02139	1
9.	Professor J. Esary NPS Code 55 Naval Postgraduate School Monterey, CA 93940	1

Thesis
S571925 Smart 193785
c.1
Analytic performance
modeling of concurrent
computer systems by
stochastic petri nets.